AFRL-RI-RS-TR-2007-241
Final Technical Report
November 2007

A FRAMEWORK FOR BUILDING AND REASONING WITH ADAPTIVE AND INTEROPERABLE PMESII MODELS

Charles River Analytics, Inc.

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

NOTICE AND SIGNATURE PAGE

AFRL-RI-RS-TR-2007-241 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/s/	/s/
DALE RICHARDS Work Unit Manager	JAMES W. CUSACK Chief, Information Systems Division Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*
NOV 2007

2. REPORT TYPE
Final

3. DATES COVERED *(From - To)*
May 06 – May 07

4. TITLE AND SUBTITLE

A FRAMEWORK FOR BUILDING AND REASONING WITH ADAPTIVE & INTEROPERABLE PMESII MODELS

5a. CONTRACT NUMBER
FA8750-06-C-0076

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
62702F

6. AUTHOR(S)

John Langton and Subrata Das

5d. PROJECT NUMBER
558S

5e. TASK NUMBER
CP

5f. WORK UNIT NUMBER
E4

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Charles River Analytics, Inc.
625 Mount Auburn St.
Cambridge MA 02138

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFRL/RISE
525 Brooks Rd
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-RI-RS-TR-2007-241

12. DISTRIBUTION AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED. PA# WPAFB 07-0301

13. SUPPLEMENTARY NOTES

14. ABSTRACT
A framework for the development and integration of Political, Military, Economic, Social, Information, and Infrastructure (PMESII) models in support of a Commander's Predictive Environment (CPE) was investigated and developed. Key challenges were allowing analysts to share their models, interconnect their models, and explore the relationships between elements in a network of integrated models. Areas explored included how to translate between different interfaces, ontologies, sub-domains, and formalisms, e.g., Bayesian networks or causal diagrams for representing and executing or simulating models. Modeling paradigms investigated included, causal graphical models, stability and reconstruction operations models (SROM), concept graphs, concept maps, and various tools for building neural network models and production rules. Also included are results of investigations, recommendations for resolving model incompatibilities, methods for reasoning with heterogeneous networks of models, and a system design specification.

15. SUBJECT TERMS
PMESII, Framework, Interoperability, Modeling, Model Integration, Commander's Predictive Environment (CPE)

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON Dale Richards
a. REPORT	b. ABSTRACT	c. THIS PAGE			19b. TELEPHONE NUMBER (*Include area code*)
U	U	U	UL	122	N/A

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY OF ABBREVIATIONS

API	Application Programming Interface
BBN	Bayesian Belief Network
BN	Bayesian Network
BNet	Bayesian Network (software tool)
CCRP	Command and Control Research Program
CI	Context Independent
CLIPS	C Language Integrated Production System
CMYK	Cyan, Magenta, Yellow, Key (black)
COA	Course Of Action
COTS	Commercial Off The Shelf
CPE	Commanders Predictive Environment
CPEF	Commanders Predictive Environment Framework
CPT	Conditional Probability Table
DAML	DARPA Agent Markup Language
DBN	Dynamic Bayesian Network
DL	Description Logic
DOM	Document Object Model
EBO	Effects Based Ontology
EBR	Evidence Based Research (Inc.)
FMF	Framework from SAIC
FSM	Finite State Model
GOMS	Goals, Operators, Methods and Selection
GOTS	Government Off The Shelf
GRADE	Graphical Agents Development Environment
GUI	Graphical User Interface
HMM	Hidden Markov Model
IBC	Integrated Battle Command
IDE	Integrated Development Environment
IHMC	Institute for Human and Machine Cognition

I/O	Input/Output
ISSM	Interim Semi-static Stability Model
JDOM	Java-based Document Object Model
JFC	Joint Forces Commander
JFACC	Joint Forces Air Component Commander
JWNL	Java WordNet Library
NPL	Natural Language Processing
OOTW	Operations Other Than War
OWL	Web Ontology Language
OWL-DL	OWL Description Logic
PIM	Political Influence Model
PMESII	Political Military Economic Social Infrastructure Information
POS	Part of Speech
RDF	Resource Description Framework
RDFS	RDF Schema
RGB	Red, Green, Blue
RLHM	(AF)RL Health Model
SDF	Synchronous Data Flow
SHIQ	A particular Description Logic
SHOIN	A particular Description Logic
SOA	Service Oriented Architecture
SPARQL	Simple Protocol And RDF Query Language
SQL	Standard Query Language
SROM	Stability and Reconstruction Operations Model
TBM	Transferable Belief Model
XML	eXtensible Markup Language
XSD	XML Schema Definition
XSL	eXtensible Stylesheet Language
XSLT	eXtensible Stylesheet Language Translations
YAML	YAML Ain’t Markup Language

SUMMARY/ABSTRACT

This report documents the development of a methodological framework for the construction, integration and exploitation of Political, Military, Economic, Social, Information, and Infrastructure (PMESII) models to infer system dynamics and provide decision support. This includes results of investigations, recommendations for resolving model incompatibilities, methods for reasoning with heterogeneous networks of models, and a system design specification. Hereafter in this report, this methodological framework will be referred to as the *Commanders Predictive Environment Framework* (CPEF). CPEF provides for interoperability of PMESII models to facilitate the re-use of legacy models and sharing and dissemination of new models. The prime candidate program for transitioning CPEF is the *Commander's Predictive Environment* (CPE). The objective of the CPE program (Carozzoni, 2005) is to build a decision support environment for the JFC/JFACC and staff to better understand the mission space (past, present & future) and predict enemy intent, actions, and emerging threats in Joint Operations. CPEF supports this objective by allowing users to collaborate, extend existing models, and reason with networks of heterogeneous models, each of which may have particular strengths in modeling some specific PMESII domain. This provides for speculative ("what-if") analysis and inference that helps predict enemy actions and responses in the overall social-political context of future military operations.

Our primary approach for CPEF centered on graphical models, given their capability of providing human-centric, easy-to-understand visual representations of relationships amongst the models' variables, along with specific underlying procedural interpretations for query execution to understand dependencies among variables. We developed methods for building such graphical PMESII models, along with a methodology for wrapping existing models to facilitate interoperability. The net result provides a strong foundation for an integrated development environment (IDE) for producing networks of interoperable PMESII models.

The candidate PMESII graphical modeling paradigms reviewed during our investigations included Ptolemy models, causal graphs, concept graphs, concept maps, and semantic and neural networks. These paradigms have been developed for a variety of purposes: some have been developed primarily for structured knowledge acquisition with minimal inferencing capabilities, some are mere black boxes used to study relationships between arbitrary input-output variables, and some offer sophisticated inferencing schemes to study complex relationships amongst their variables. But each of these modeling paradigms allows some form of speculative, or what-if, analysis to explore the static and dynamic effects of one subset of model variables on another such subset, within the target application domain (along a PMESII dimension) for which the

model was built. Therefore, when a network of models is built from a number of different PMESII models incorporating a variety of different technologies, the result is a system with powerful capabilities for exploring relationships among variables spanning several of the encompassed domains.

Our efforts leveraged existing models and tools for various modeling paradigms, including our in-house tool, BNet.Builder™ for building causal graphical models, government-off-the-shelf (GOTS) stability and reconstruction operations models (SROM) from AFRL/RI, commercial-off-the-shelf (COTS) tools such as VivoMind© toolset for concept graphs and CmapTools© for concept maps from our strategic partners, and various open source software packages for building neural network models and production rules. Development of the aforementioned IDE itself, based on our in-house tool, GRADE™, was researched under AFRL contract FA8750-06-C-0078, *Toolkit for Building Hybrid, Multi-Resolution PMESII Models*, and is documented in that contract's final report. The GRADE-based IDE was used as an enabling test-bed for the methodological framework developed under this effort.

1. Introduction

1.1 Problem Statement

Modern warfare has become far more complex than traditional force-on-force warfare. Some causes of this complexity include: the adversary's increasingly skillful use of the indigenous population and infrastructure, coordination of global war efforts involving multiple and heterogeneous coalition partners (each with various political and other constraints), and the impact of friendly and adversary actions on, for example, the economic, political, and social dimensions of all countries involved, whether adversary, ally, or "interested neutral." There is thus a clear need to assist the commander in understanding the structure and dynamics of this complex battlespace that spans and interconnects such diverse dimensions, and that makes forecasting and prediction such an immensely difficult task. It is clear that no analysis done in isolation along a *single* dimension, i.e., not taking into account the effects of the other dimensions, can give a full picture of the battlespace in its socio-political context. Nonetheless, there are many ongoing efforts to develop and field isolated tools and models along single Political, Military, Economic, Social, Information, and Infrastructure (PMESII) dimensions. What is needed to make full use of these efforts is an effective integration and exploitation of these models for an in-depth and holistic understanding of the military situation within the socio-political context. This report presents CPEF, our methodological framework for such integration and exploitation. CPEF adapts existing models to ensure interoperability, facilitates model integration, and supports reasoning with heterogeneous networks of models to more accurately capture battlespace dynamics.

Here we report the results of our efforts in developing CPEF to support analysts integrating a network of interoperable PMESII models, and how we can exploit this network of models to explore dependencies and influences amongst selected battlespace variables. Consider the process of building individual models and integrating them into a network of interoperable models: it can be viewed as a joint activity between human analysts and subsystems implementing particular graphical modeling paradigms. The analyst will interact with the appropriate subsystem to produce a model in his particular area of expertise, incorporating his choice of modeling paradigm. For example, a subsystem for constructing causal graphs may be the best choice for representing probabilistic relationships among variables from the domain of terrorism and insurgency.

The collection of PMESII models so constructed will result in an inclusive global semantic network containing a set of interconnected ontological terms and phrases capturing both static and dynamic properties of, for example, a terrorist organization of interest, including their structures, associations among members, influence towards each other, cultural and behavioral profiles of members, and their social and economic networks. We may then exploit this network in various ways, e.g., we may speculatively determine the effects of interrupting the economic support of the terrorists, perform situation assessments, or even predict such conditions several months hence.

1.2 Objectives

The primary objective of the CPEF effort was to develop a methodological framework for building networks of integrated, interoperable, and adaptive PMESII models for enhanced understanding of the mission space. Specific objectives were:

- **Scenario and Scope Definition:** We worked with subject matter experts to develop scenarios with sufficient complexity to demonstrate the basic functionalities of the CPEF framework. We then defined the scope and requirements, guided by the chosen scenarios. This included specifying the software and hardware requirements and the bounds on functionality for the prototype, specifying the range of decision support functionality for the commander and staff that should be supported in the prototype to better understand the mission space, requirements for interoperability with existing models, and requirements for inferencing. We then generated a functional specification detailing key implementation issues.
- **PMESII Models Development:** We developed a methodology for skilled/non-skilled analysts to rapidly build various types of graphical PMESII models capturing key relationships, dependencies, and vulnerabilities of adversary/self/neutrals. Our initial focus was on building models in the context of the chosen scenarios using a variety of graphical modeling paradigms such as concept graphs, causal graphs, concept maps, influence diagrams, and neural networks. We leveraged existing commercial-off-the-shelf, in-house, and open source tools that exist for some of these modeling paradigms.
- **Reasoning with Models and Adaptation:** We developed inferencing capabilities for a network of PMESII models to identify leverage points that represent opportunities to influence capabilities, perceptions, decision making, and behavior. Inferencing was developed for reasoning with information flowing 'between' models (meta-level or inter-model) and within models (intra-model). The latter category of inference techniques includes graph matching, analogical reasoning, evidence propagation, and arc traversal to cover the graphical modeling paradigms. We leveraged existing inferencing techniques wherever possible. We investigated enhancing the inference techniques to include spatiotemporal relations among models and variables. We also studied appropriate machine-learning techniques for adapting individual models over time.
- **Making Models Interoperable:** We devised a methodology that enables the construction and integration of interoperable PMESII models. The models were drawn from existing models with interface "wrapping" added for interoperability, along with models developed from scratch. This task addressed the incompatibilities and gaps identified in Section 2.1 (and illustrated in Figure 1).

- **Demonstration of Proof-of-Concept:** We built specific graphical models in the context of the battlespace scenario developed under the scenario definition task above, involving multiple PMESII dimensions to carry out mission specific tasks. The model building process and reasoning functionality was prototyped and tested using our in-house GRADE-based IDE.
- **Prototype Validation and Evaluation:** We developed a validation plan and evaluated the limited-scope prototype. This was done in terms of assessing its accuracy in identifying potential semantic relations between models to support model integration and ultimately reasoning with networks of heterogeneous models.
- **Transition and Final Reporting:** We identified transition opportunities and developed a plan to exploit the results of our CPEF work. Results of the CPEF effort are summarized in this document and include recommendations for integrating and reasoning with PMESII models as well as a system design specification that can be used for implementations of a full-scope prototype.

1.3 Report Outline

Section 2 describes some of the key issues and challenges in developing a methodology and framework to support construction, integration, and reasoning with a network of heterogeneous PMESII models. This includes an enumeration of potential model incompatibilities and functionality gaps in Section 2.1. Section 3 presents a record of our performance in the form of a project task breakdown. Section 4 provides general recommendations for a framework that supports PMESII model construction, adaptation, and integration. These recommendations are relevant to the entire CPEF system and simultaneously apply to more than one of the incompatibilities and functionality gaps presented in Section 2.1. Also included is a discussion of what constitutes an ontology, issues concerning the use of ontologies, and how the CPEF methodology and framework leverages different levels of ontology specification but is not restricted by a lack of ontologies. Section 5 offers recommendations that are specific to each incompatibility and functionality gap presented in Section 2.1. We provide a system design specification in Section 6 which describes the components and processing stages of CPEF and lists enabling technologies that can be used for the implementation of each component in a full-scope prototype. We conclude with Section 7 which discusses methods for reasoning with a network of heterogeneous PMESII models and the issues involved.

2. Key Issues and Challenges in Constructing, Adapting, and Integrating PMESII Models

This section enumerates and describes issues central to constructing, adapting, and integrating PMESII models. Section 5 provides recommendations for resolving each of the issues described in this section.

2.1 Model Interoperability: Incompatibilities and Functionality Gaps

There are several fundamental issues (and associated "hard" problems) that need to be addressed in undertaking the development of an interoperable framework of PMESII models. First and foremost is the problem of making existing or even new models interoperable, as these are developed independently, i.e. with no coordination, by different software design and development teams, in consultation with domain experts having various levels of skills and expertise. A very common approach is to build a wrapper around an existing model, thus converting it to an input-output black box, or to provide an intelligent agent operating autonomously, which communicates with other models in the network. But this approach is likely to introduce other types of gaps and incompatibilities between models, some of which are illustrated in Figure 1. The major thrust of our effort was to identify an overall methodology to fill these gaps, including various intelligent automated techniques, processes and guidelines, and aid from human subject matter experts and analysts wherever needed.

2.1.1 Interface Incompatibility

The first problem shown in Figure 1 (in the top row) concerns *interface incompatibility* between two PMESII models that either already exist or are being developed independently. If we intend to feed output from model A about a certain object X as input to model B, then some *mismatch* between the output and input may occur in terms of the assumptions about the numbers and types of X's attributes. This is often straight-forward but tedious to deal with, often merely involving "translation" from one descriptive framework to another, e.g., from numerical values - 1, 2, 3, … - to "fuzzy" values (low, medium, high, …). A bigger problem ensues when different levels of resolution are used to represent the same object in two different models. If model A provides a high-resolution object representation of X, e.g., a map, or enemy force estimates, for model B, and model B needs a low-resolution representation, e.g., lat/long coordinates, or enemy center of gravity, then some aggregation process must be conducted, usually based on one approximation method or another. The reverse process is much more difficult, going from a low-resolution output to a high-resolution input, since, in effect, missing input attributes have to be inferred/approximated and "filled in."

CPEF uses a number of approaches to resolve the interface incompatibility. These are described in Section 5.1. Recommendations are also made for more comprehensive approaches which can be implemented in a full-scope prototype.

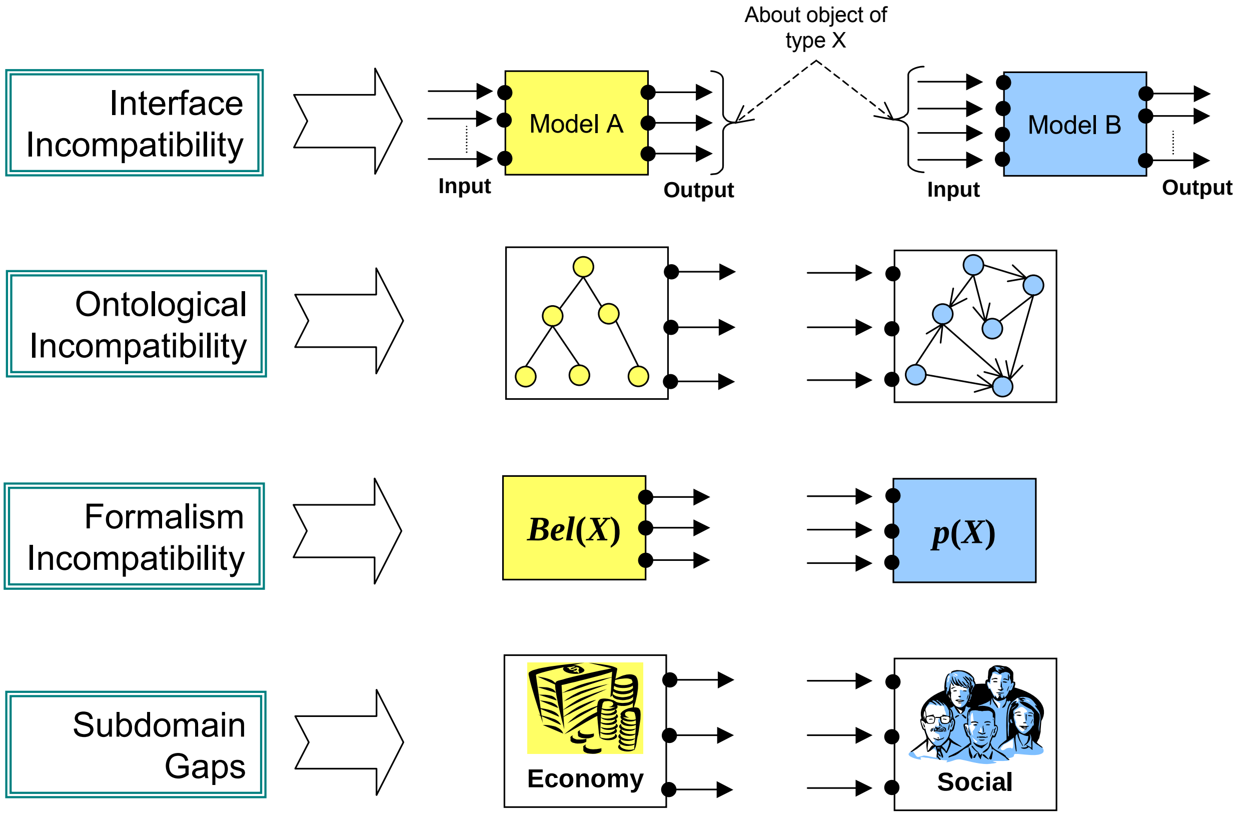


Figure 1: Gaps and Incompatibilities Between PMESII Models

2.1.2 *Ontological Incompatibility*

The second problem illustrated in Figure 1 concerns *ontological incompatibility* between models that arises due to differing vocabularies and expressive power in their respective ontologies. Different teams of engineers and subject matter experts with a diverse range of expertise, knowledge, and cognitive capabilities independently creating PMESII models will inevitably develop and use different underlying ontologies; which, in turn, will give rise to incompatibilities across models. Initially, one might suggest the development of a "common ontology" for the set of all possible PMESII models; but many failed efforts in this direction make it clear that developing a universal ontological standard for model creation is impractical, if not theoretically impossible. Moreover, if models are to be built rapidly, analysts should ideally be free to use a model-building environment of their own choosing without assistance from knowledge engineers, and thus the analysts should not be constrained by a predefined ontology to express their knowledge, which usually inhibits their expressive flow. Hence, rather than proposing to develop a common ontology for the PMESII model space, CPEF focuses on facilitating better mapping capabilities between differing ontologies. For example, there are tools that can map ontological terms from one domain to another by solving the well-known synonymy and polysemy problems; these clearly offer hope for translating between differing ontologies used in the models. In some cases of incompatibility between the underlying ontological structures of the models, e.g., semantic networks vs. logical expressions, one domain can be mapped to another by providing a more expressive ontological structure for one of the models, e.g., semantic networks can be mapped to first-order logical sentences. Therefore, some parts of the ontological incompatibility problem can be addressed via automated techniques, which we have explored and leveraged under the CPEF effort.

CPEF uses a number of approaches to resolve the ontological incompatibility. These are described in Section 5.2. Recommendations are also made for more comprehensive approaches which can be implemented in a full-scope prototype.

2.1.3 Formalism Incompatibility

While ontological incompatibility creates problems due to multiple ways of designating an entity, the *formalism incompatibility* shown in Figure 1 is concerned with multiple ways of instantiating the object entity computationally represented in the model. For example, uncertainty can be expressed not only in terms of probability values, but also via various other formalisms such as certainty factors, the Dempster-Shafer measure of beliefs, and numerous other qualitative dictionaries. These are fundamentally incompatible with each other, both in terms of their underlying conceptual representation of uncertainty and probabilistic reasoning, as well as in the sense of having different types of scales. Conversion between two such formalisms often requires deep understanding of the models and their formalisms, thus breaking the simple input-output black box idea of encapsulation. Specialization of formalism is often appropriate to map one approach to another. For example, probability theory is a special case of Dempster-Shafer theory that allows beliefs to be expressed only on singleton sets, and so facilitates development of a mapping from probability models into Dempster-Shafer models.

CPEF uses a number of approaches to resolve the formalism incompatibility. These are described in Section 5.3. Recommendations are also made for more comprehensive approaches which can be implemented in a full-scope prototype.

2.1.4 Subdomain Gaps

Finally, if one wants to feed the output from a model in one PMESII domain to another, it will require an analyst or domain expert with knowledge of both domains to bridge the *subdomain gaps.* This is due not only to the ontological gaps between the domains being considered, but also to differing dynamics between the domains. Addressing this problem requires the skills of experts from the respective domains, or ideally those who are expert in both domains.

CPEF facilitates experts bridging such gaps by highlighting possible correspondences between concepts and variables across domains. A number of approaches are proposed to resolve the issue of subdomain gaps. These are described in Section 5.4. Recommendations are also made for more comprehensive approaches which can be implemented in a full-scope prototype.

2.2 Reasoning with PMESII Models and Adaptation

Automated reasoning with PMESII models helps analysts understand key relationships, dependencies, and vulnerabilities of adversary/self/neutrals, and identifies crucial leverage points that represent opportunities to influence various capabilities. We envision three types of reasoning with the models: deductive, abductive, and inductive. *Deductive reasoning* enables analysts to determine influence of one variable onto others. For example, variables representing the supply of electricity from an infrastructure model influence the variable representing the level of anger among a population in a social model. Conversely, *abductive or explanation based reasoning* allows analysts to determine the variables that affect, or explain, a variable of interest, in order of their influence. For example, the variable representing the supply of fresh water may be the most influential in explaining the anger among certain sections of a population. *Inductive reasoning* finds patterns in observations and converts them to knowledge. So if we repeatedly find that electrical shortages induce angry demonstrations among certain sections of a population, then by induction this pattern of influence becomes knowledge, and the underlying model can be enhanced by incorporating the newly learned pattern. We have leveraged our in-house expertise of various symbolic, linear and non-linear learning and pattern recognition techniques to implement the necessary inductive reasoning for the CPEF effort.

While the unidirectional flow of information within a component-oriented, input-output type of black box model is a natural fit for deductive reasoning, it is much harder to perform abductive or explanation based reasoning. This type of "backward" reasoning typically requires computation of influence backwards, which is especially difficult when input-output relationships are not easily explainable or invertible, e.g., neural networks or differential equations. Infinite combinations of values of input variables that come from real-valued domains prevent application of a trial and error method to determine optimal explanations (as we cannot generally expect such functions to be invertible). In the specific discrete case of reasoning with graphs in general, we addressed this question by assessing the utility of various graphical model specific inferencing techniques, for example, analogical reasoning for concept graphs, evidence propagation for causal graphs and influence diagrams, adaptation and back propagation for neural network models, and graph matching techniques for semantic networks in general. Because we are emphasizing graphical models, these techniques have the potential of solving a majority of the difficulties associated with the abductive reasoning techniques we considered for inclusion in CPEF.

Adaptation of a graphical PMESII model involves changing or tuning its structure to produce behavior in line with past observations. Adaptation of computational models in general is typically achieved via a variety of machine learning algorithms. Changing the structure of a graphical model requires addition/deletion of its nodes representing concepts and associated links defining their relationships. Our network/component-oriented approach to building interoperable models allows us to achieve adaptation of a network in terms of local adaptations of the individual models it contains. For instance, we can leverage existing learning algorithms to adapt individual modeling paradigms. For example, there are established learning algorithms for adapting or learning from scratch the structure and probabilities for a causal graph. Similar algorithms exist for production rules. For neural network models, adaptation of a network requires its retraining. Semantic graphs in general require human intervention for adaptation.

In Section 4.3.12 of this report we review the types of reasoning possible with different levels of ontology specification. We also address implications for inference using networks of heterogeneous PMESII models in Section 7. While a primary focus of our effort has been on model construction, integration, and adaptation, once these principle functionalities are provided, many of the issues in reasoning with models will be addressed. For instance, the issue of how to reason with a Bayesian network and rules base, and specifically the problem of propagating evidence from a Bayesian network to a rules engine, will be addressed when the interface incompatibility described in Section 2.1.1 is addressed.

2.3 Building Graphical PMESII Models

If we view PMESII models as abstract computational representations of domain experts' knowledge, then building such a model from scratch is typically a time-consuming process involving consultation with the appropriate domain experts. Semantic networks provide us a means of reducing the effort in that process since they are an especially effective means of representing an analyst's understanding of a particular domain of knowledge. In general, there is no specific formal semantics for semantic network representations, that is, there is no agreedupon notion of what a given representational structure means, as there is in mathematical logic or Bayesian networks.

2.3.1 Semantic Networks

Semantic networks consist of nodes and links between the nodes. The nodes represent objects or concepts and the links represent relations between nodes. The links are directed and labeled, and thus a semantic network is a directed graph. In the simplest form of a semantic network circles or boxes usually represent the nodes and the links are drawn as arrows between the circles. The network defines a set of binary relations on a set of nodes. Now to be useful as a modeling tool, semantic networks must obviously facilitate processing of the information they contain. The way that the information is processed is known as arc traversal. This is used to identify the complex relationships between nodes.

The following are six of the most common categories of semantic networks (Sowa, 1991):

- *Definitional* networks emphasize the subtype or 'is-a' relation between a concept type and a newly defined subtype. The resulting network supports the rule of inheritance for copying properties defined for a supertype to all of its subtypes. The traditional hierarchy within an object-oriented framework falls into this category.
- *Assertional* networks are designed to assert propositions. Unlike definitional networks, the information in an assertional network is assumed to be contingently true.
- *Implicational* networks use implication as the primary relation for connecting nodes. They may be used to represent patterns of beliefs, causality, or inferences. Bayesian belief networks are implicational networks.
- *Executable* networks include some mechanism, such as marker passing or attached procedures, which can perform inferences, pass messages, or search for patterns and associations. Markov decision processes can be represented as executable networks.
- *Learning* networks build or extend their representations by acquiring knowledge from examples. The new knowledge may change the old network by adding and deleting nodes and arcs or by modifying numerical values, called weights, associated with the nodes and arcs. Neural networks are examples of learning networks.
- *Hybrid* networks combine two or more of the previous techniques, either in a single network or in separate, but closely interacting networks.

In the CPEF effort we reviewed and evaluated a number of well-known graphical modeling paradigms from the above types of semantic network categories for building PMESII models including concept maps, concept graphs, causal graphs, influence diagrams, and neural networks. Each of these paradigms has its own strengths and weaknesses, thus making each suitable for some PMESII applications but not all. In Table 1 we list a number of the candidate modeling paradigms reviewed along with a comparison of their relative strengths and weaknesses.

2.3.2 Candidate PMESII Modeling Paradigms

Under the CPEF effort we constructed networks of PMESII models using a variety modeling formalisms including causal graphs, concept graphs, concept maps, semantic networks, neural networks, SRO models, and production rules. This was to determine what PMESII sub-domains each was most appropriate for and to facilitate scenario construction and CPEF prototype validation and evaluation. Table 1 compares these modeling formalisms along a number of dimensions. We define these dimensions below and briefly describe each of the candidate modeling formalisms.

Table 1: Comparison of Candidate PMESII Model Formalisms

Models and Characteristics	Expressivity	Executable	Reasoning	Adaptability	Tools
Concept Map	High	No	Forward Backward	Medium	COTS Free (Research)
Concept Graph	Medium	No	Forward Backward	Low	COTS Free (Research)
Causal Graph	Medium	Yes	Forward Backward	Medium	In-House COTS Free (Limited)
SRO Model	Medium	Yes	Forward Backward	Low	GOTS
Neural Network	Low	Yes	Forward	High	COTS Open Source
Production Rule	Medium	Yes	Forward	Low	COTS Open Source

Expressivity of a modeling formalism refers to its ability to capture and express an analyst's knowledge in terms of the constructs the formalism offers. The expressivity of a concept graph is very high as it keeps the phrases used by the analysts intact in the model. In contrast, a neural network model is only able to keep the input-output relationships in the model.

The e*xecutable* characteristic of a modeling formalism refers to whether some useful information that is implicit in a model, e.g., degree of influence of one variable onto another, can be derived from the model via some kind of inferencing algorithm. A causal graph, for example, is an executable formalism as it offers propagation algorithms, and so also is a trained neural network. In contrast, the concept mapping model does not have such an algorithm. But a graphical model will always be able to determine the importance of a variable based on the number of incoming/outgoing edges from the variable, or detect the influence of one variable on another based on interconnections (although not necessarily quantifying the strength of the influence).

Reasoning of a modeling formalism refers to the formalism's ability to detect the direction of influence (not just connection) of one variable to another. A belief network propagation algorithm, for example, incorporates both deductive and abductive reasoning, and thus able to detect both forward and backward influences along with their degrees. On the other hand, the standard back propagation neural network modeling formalism is limited only to forward reasoning.

Adaptability of a modeling formalism refers to automatic adjustments by models, which are necessary to take into account new observations. It is hard to adjust structures of graphical models as they are built in consultation with subject matter experts. But the strength of relationships among a set of variables within a model, e.g., probabilities in a belief network model or activation levels within a neural model can be adjusted based on observations without changing their structure.

Tools of a modeling formalism refers to the currently available software tools implementing the formalism, that is, whether such a tool is in-house, COTS, GOTS, open source, or freely available for research/commercial purposes.

We now briefly describe the different contemplated modeling techniques.

2.3.2.1 Concept Maps

Concept maps are a result of research into human learning and knowledge construction (Novak, 1998). In concept maps the primary elements of knowledge are concepts, and relationships between concepts are propositions. Concept maps are a graphical two-dimensional display of concepts (usually represented within boxes or circles), connected by directed arcs encoding brief relationships, e.g., linking phrases, between pairs of concepts forming propositions. Each concept node is labeled with a noun, adjective or short phrase, and each edge is labeled with verbs or verb phrases describing the relation between the connected concepts. Concepts maps are highly effective in quickly capturing domain knowledge along PMESII dimensions.

Concept maps are organized in a somewhat hierarchical structure, with the more general concepts at the top of the graph and the more specific concepts nearer the bottom. The cardinality of the relationships is unrestricted; however, most relationships either connect one concept to another single concept, or connect one high-level concept to several more specific concepts. This generally leads to diagrams with a tree-shaped structure, although this is not mandatory. Figure 2 shows a typical concept map.

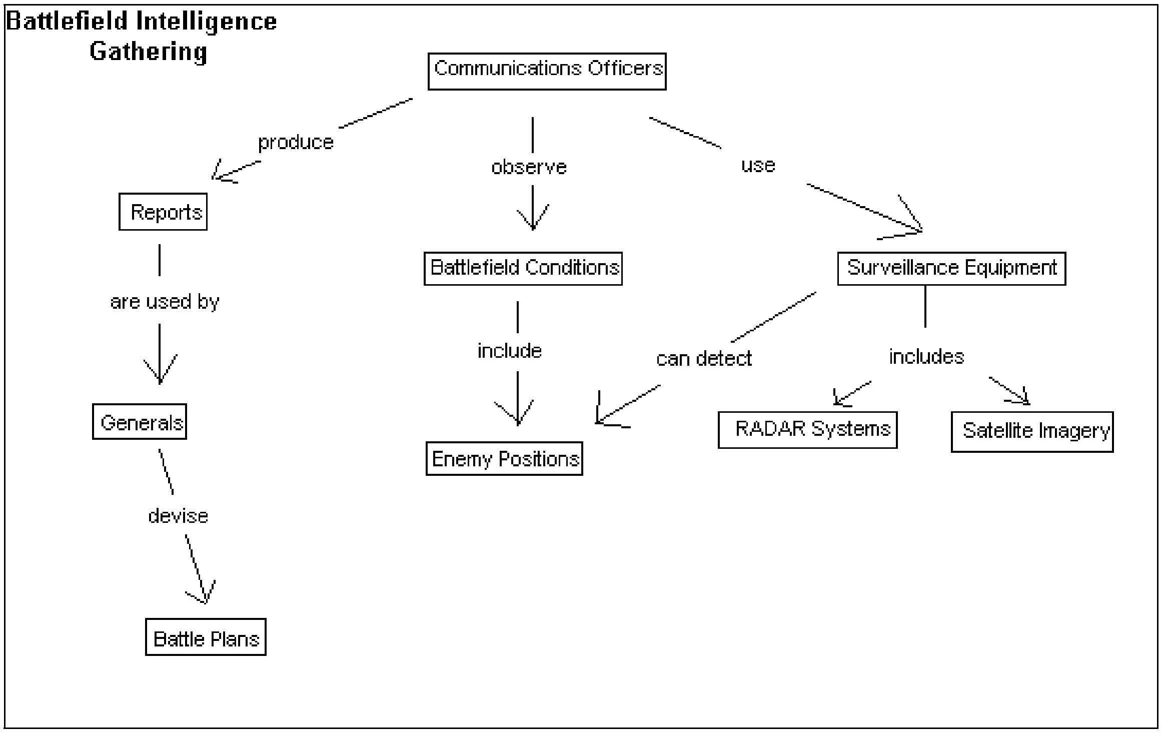


Figure 2: Concept Map Describing Battlefield Intelligence Gathering

In Figure 2 we see a cognitive map diagram describing the process of battlefield intelligence gathering. Note that the map here is not a strict hierarchy. The “can detect” relationship between the concepts “Surveillance Equipment” and “Enemy Positions” is an example of a crosslink, or a relationship that joins different map segments.

Often, the process of creating a concept diagram is of equal or greater utility than the finished diagram itself. As a collaborative process, concept mapping can help an interviewer elicit details of a complicated topic from one or more domain experts. The interviewer starts with a focus question, and asks the domain experts for a list of concepts relevant to this question. This list should be large enough to answer the focus question in satisfactory detail, but should be no larger than the number of concept nodes that will fit in a single diagram (usually no more than 40 or 50). The interviewer creates concept nodes to represent these concepts, using either sticky notes placed on a whiteboard or a software package designed for cognitive mapping. The interviewer then asks the experts to group related concepts on the diagram, placing the more general or central concepts near the top of the diagram, and defines the relationships between the related concepts. During this process it is common for the domain experts to debate the precise nature of the relationships between the concepts, and clarify and revise the diagram accordingly. Concepts may be added or removed during this phase as necessary. Finally, the diagram is examined once more to make sure that all parties present are satisfied that the diagram accurately answers the focus question.

A popular tool for concept mapping is the CmapTools (Canas et al., 2004) package developed at the Institute for Human and Machine Cognition (IHMC) (www.ihmc.us). The package is freely available for both commercial and non-commercial use, and has many advantages over using sticky notes or a more general diagramming tool, e.g., it can record the entire mapmaking process.

2.3.2.2 Concept Graphs

Concept graphs are a formal system of logic based on the existential graphs of C. S. Peirce and semantic networks. They are mathematically precise and computationally tractable structures, which have a graphic representation that is humanly readable. For this reason, concept graphs have been used in a variety of applications for computer linguistics, knowledge representation, information retrieval, and database design. Figure is an example concept graph encoding a generic behavioral model of a terrorist leader.

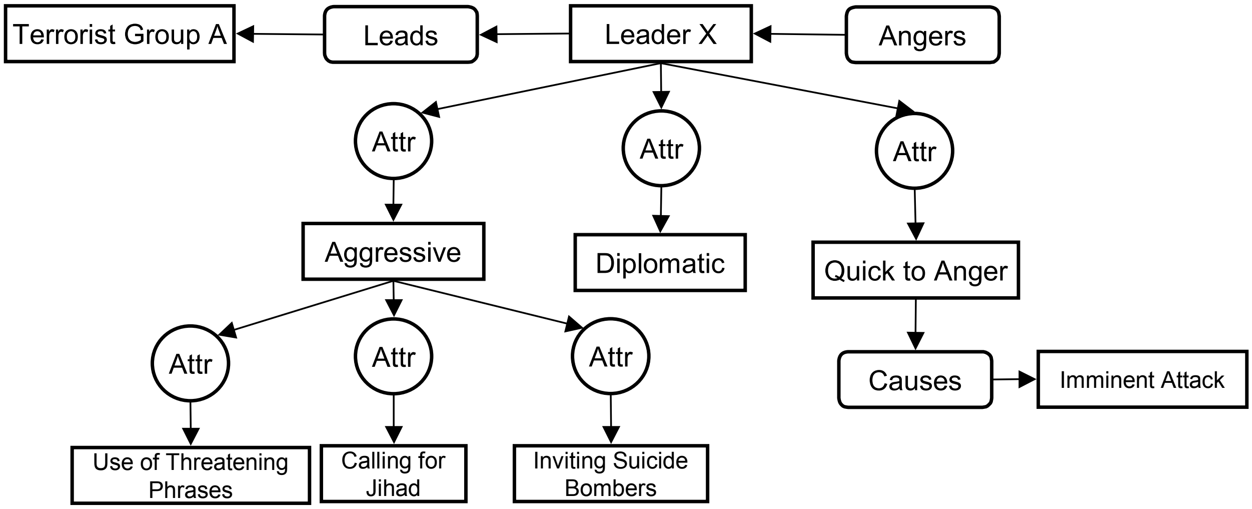


Figure 3: Concept Graph Model for Terrorist Leader Behavior

In concept graphs, concepts are represented by rectangles, e.g., [Person: Leader X] and [Behavior: Aggressive], and conceptual relations are represented by circles and ovals, e.g., Leads, Causes. An analyst can query such a model to determine who the terrorist leader is and the nature of the leader, based on various observable intelligence. Such a leader X, who leads the terrorist group A, can possess different types of behavior attributes including aggressive, diplomatic, quick to anger, etc. If such a leader is quick to anger and there are some stimuli to make the leader angry, then an attack on friendly targets may be imminent.

Under the CPEF effort we used the VivoMind© toolset for concept graphs from our strategic partner, SAIC. VivoMind© toolset provides a modular, semantic service-oriented framework, called FMF, encapsulating powerful analogical reasoning capabilities based on concept graphs to create analogically reasoning agents. These agents can find analogies and provide explanations so easy to understand, yet non-obvious: they can discover significant new features in input information and ontologies automatically. They also work to provide valuable control knowledge to other reasoning modules and to thereby enable agile model-building of the problem domain. The tools that SAIC, Inc. has developed using the FMF approach are capable of highly efficient and effective ontological reasoning based on conceptual graphs. An ontology constructed with these tools is then a collection of types and individuals, which forms a framework for the knowledge in a domain. The collection is arranged into a hierarchy based on subtypes and generalization and specialization. Every concept on the hierarchy is treated as a type, where there is no longer a need to distinguish instances from types. The essential difference is in, for example, treating a freighter as you would any generic ship. The type ship can be placed, occupy space, and have specific values for weight or direction. The generic ship can have constraints placed on its attributes, and finally can be specialized into a freighter when enough data has been gathered to support this hypothesis.

2.3.2.3 Causal Graphs

A *causal graph* (a.k.a. belief network) (Jensen, 1996) is a graphical, probabilistic knowledge representation of a collection of variables describing some domain, yielding a PMESII model. The nodes of the network denote the variables and the links denote causal relationships between the variables. The topology encodes the *qualitative* knowledge about the domain. Conditional probability tables encode the *quantitative* details (strengths) of the causal relationships.

The belief network of Figure 4 encodes some aspects of a social model (disregard the diamond box for now). The top half of the network captures the influence of four variables, namely, *power*, *drinking water*, *refined fuel*, and *sufficient food supply*, on a variable representing the level of anger of the population in a town aligned with coalition forces. The dynamics of the social model are that short supply in any one of these three consumable products will increase the level of anger among the local population. The bottom half of the network encodes the effects of an angry population in terms of street protests and ambushes on the patrolling forces. A probability table specifies the probability of each possible value of a child variable, conditioned on each possible combination of its parent variable values. For example, the probability of street protests given that the anger level is high is 0.9.

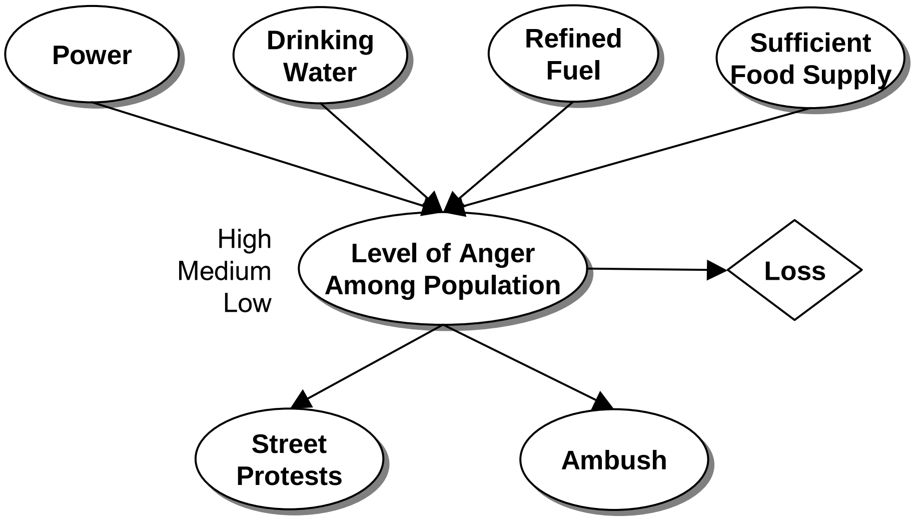


Figure 4: Simple Causal Model and Influence Diagram

The chance that a node of a causal graph is in a particular state is termed its 'belief' in the state, reflecting the probability that the node is in that state given all the previous evidence received. The structure of a belief network encodes other information as well. Specifically, the lack of links between certain variables represents a lack of direct causal influence, that is, they indicate conditional independence relations. This belief network encodes many conditional independence relations, for example,

$$\textit{Street Protests} \perp \textit{Ambush} \mid \textit{Level of Anger Among Population}$$

$$\textit{Ambush} \perp \{\textit{Power, Drinking Water}\} \mid \textit{Level of Anger Among Population}$$

where '⊥' is read 'is independent of' and '|' is read 'given.' That is, once the level of anger among the population is known, the observation of street protests adds no further information about the likelihood of ambush. Similar conditional independence assertions hold for the other variables.

When new evidence is posted to a variable in a causal graph, that variable updates its own belief vector and then sends out messages indicating updated predictive and diagnostic support vectors to its children and parent nodes respectively. These messages are then used by other nodes to update their belief vectors and to propagate their own updated support vectors. The separation of evidence yields a propagation algorithm (Pearl, 1988) in which update messages need only be passed in one direction between any two nodes following posting of evidence. Thus, the algorithm's complexity is proportional to the number of links in the network. This separation also automatically prevents the possibility of double-counting evidence. Our in house tool Bnet.Builder® facilitates construction and reasoning with causal graphs.

Influence diagrams are a specialization of causal networks, augmented with decision variables and utility functions to solve decision problems. *Decision trees* are specialized influence diagrams that help to choose between options by projecting likely outcomes as utilities. The diamond box "Loss" in Figure represents the expected mission utility in line with the level of anger. The utility (though difficult to quantify here) should go up when the anger level is down and vice versa.

2.3.2.4 SRO Models

A *Stabilization and Reconstruction Operations Model* (SROM) (Robbins, Deckro, & Wiley, 2005) analyzes the organizational hierarchy, dependencies, interdependencies, exogenous drivers, strengths, and weaknesses of a country's PMESII systems to enable more efficient resource expenditure. SROM models a country in a holistic manner as a national sub-model, which is then defined in terms of its regions as a system of systems (shown in Figure). Regions can interact with each other as well as the National Sub-Model. Each regional sub-model itself contains six functional sub-models: demographics sub-model, insurgent and coalition military sub-model, critical infrastructure, law enforcement, indigenous security institutions, and public opinion. The utility of SROM is briefly demonstrated below, using

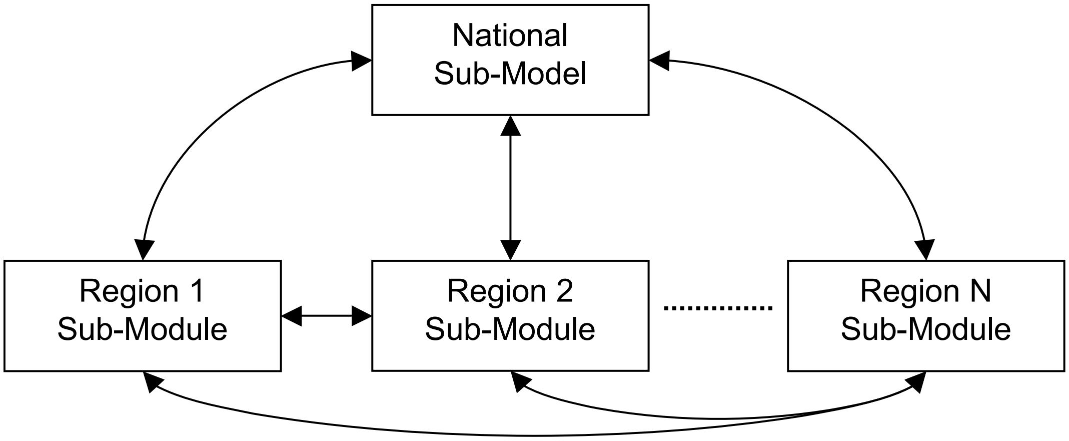


Figure 5: Top Level Nation SROM (Robbins et al., 2005)

Figure below shows a portion of the critical infrastructure model of SROM. The model captures a sequence of influences among variables, starting from the power supply at an electrical substation. The generated power is fed into an industrial water plant, which produces water consumed by oil field work. An oil field produces crude oil to be refined by a refinery. Finally, refined fuel is used to generate power, which in turn is supplied to various power substations, thus forming a loop.

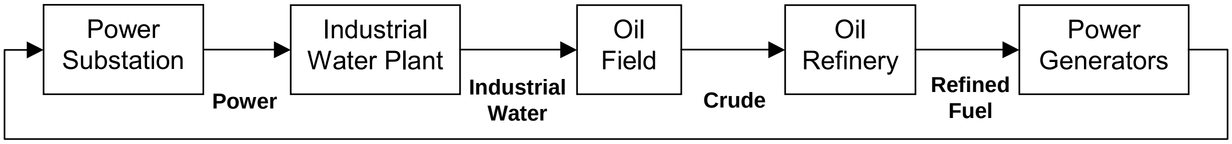


Figure 6: SROM Infrastructure Model

2.3.2.5 Neural Networks

A *neural network* is a nonlinear information-processing paradigm that attempts to mimic the functioning of the brain in a simplistic manner. The key element of this paradigm is the novel structure of the information processing system, composed of a large number of highly interconnected processing elements (a.k.a. neurons or nodes), arranged in multiple layers, working in unison to solve specific problems. A sample neural network is shown in Figure . Neural networks offer some of the most versatile ways of mapping or classifying a nonlinear process or relationship. Neural networks will help us to build PMESII models for those domains that have highly complex non-linear relationships between input and output variables. The exact relationship between input and output variables in a neural network will not be transparent due its black box nature, but the effect on output variables for various inputs can be observed via forward propagation.

Neural networks derive power and versatility mainly through the activation functions or the nonlinearity associated with a node. The inputs to a node are the available measurements or the scaled outputs from other nodes. Sigmoid activation functions are most common. Two major kinds of network topology are feed-forward and feedback. In the feed-forward neural network structure, signals flow between the nodes only in the forward direction, i.e., from the inputs towards the outputs. In a feedback or recurrent neural network, the outputs from nodes may flow in the reverse directions. Feed-forward neural networks are restricted to finite-dimensional input and output spaces. Recurrent neural networks can in theory process arbitrarily long strings of numbers or symbols, but training recurrent neural networks poses much more serious practical difficulties than training feed-forward neural networks. Multilayer perceptrons and radial basis function networks are the two most commonly used types of feed-forward networks.

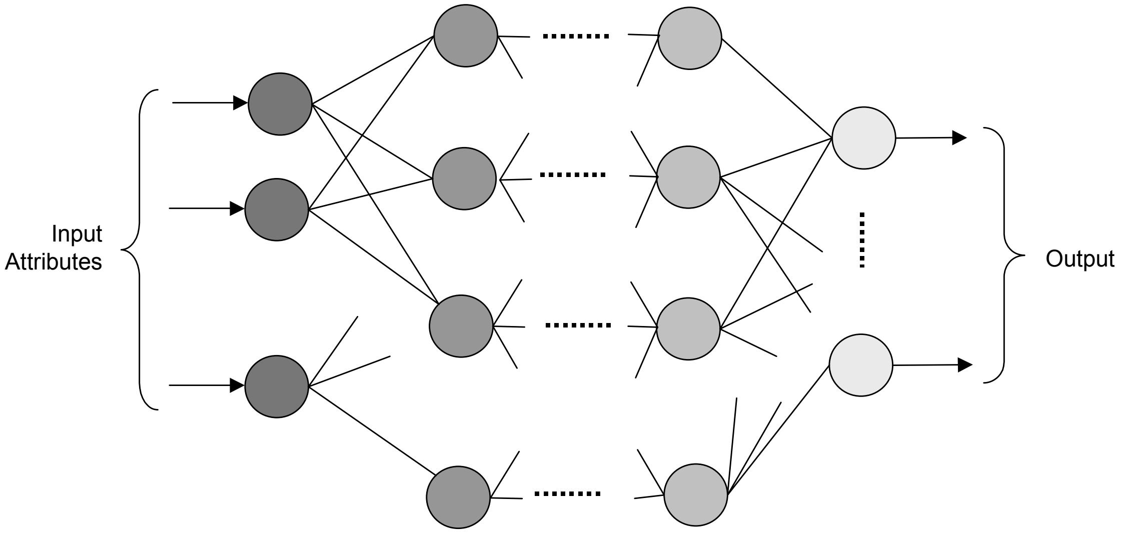


Figure 7: Example Neural Network Structure

The weights on the nodal interconnections are estimated iteratively by a nonlinear optimization method. A variety of numerical optimization methods, e.g., conjugate gradient method, can be used for training feed-forward neural networks. Conventionally, a training set of data is used for learning or parameterization of the network; a validation set of data are used for optimization of the architecture and a test set of data are used for performance assessment (Ripley, 1996).

In practice, neural networks are especially useful for classification and function approximation or mapping problems which are tolerant of some imprecision, which have lots of training data available, but to which hard and fast rules (such as those that might be used in an expert system) cannot easily be applied. Almost any finite-dimensional vector function on a compact set can be approximated to arbitrary precision by feed-forward neural networks (which are the type most often used in practical applications), with enough data and enough computing resources.

Different variations of neural networks have been developed. For example, a neural net can incorporate fuzziness or heuristics in the inputs, the outputs, or in the constraints in the nodes (Carpenter et al., 1996); in Self-Organizing Maps, competitive networks provide a topological mapping from the input space to the clusters (Kohonen, 1984). Evolutionary Neural Networks derive power from the training of the neural network from evolutionary algorithms, e.g., genetic algorithms.

Neural networks have been successfully used in diverse paradigms, for example: recognition of speakers in communications, diagnosis of hepatitis, recovery of telecommunications from faulty software, interpretation of multi-meaning Chinese words, undersea mine detection, texture analysis, three-dimensional object recognition, hand-written word recognition, and facial recognition.

2.3.2.6 Production Rules

A *production rule* (aka if-then rule) can be thought of as a simple graphical structure whose nodes are built from the conjunction of conditions in the antecedent of the rule, and then linked to another node corresponding to the consequent of the rule. Several rules can be chained together by matching the consequent of one rule with a condition of the antecedent of another rule, thus forming a larger forward-chaining graph representing a PMESII model. The kind of rules we shall be dealing with in this effort has the following general form:

IF *Events*

THEN *Hypothesis* (*D*)

The above rule is interpreted as "If *Events* occur then *Hypothesis* follows with degree of uncertainty *D*". If *D* is a probability value then the rule implies that "If *Events* occur then the probability that *Hypothesis* will follow is *D*". *Events*, the antecedent of the rule, is a conjunction of propositional symbols each representing an event and *Hypothesis*, the consequent, can be a property symbol as in

IF *Interrupted Power Supply* AND NO *Gasoline in Market* THEN
Anger among Population (0.8)

or an action symbol as in

IF *Anger among Population* THEN *Keep QRF Stand By* (0.8)

There are COTS and open source tools on the market for both deductive and abductive reasoning with production rules. For example, CLIPS is an open source free tool and ILOG is commercially available. Moreover, our in-house tool GRADE includes a built-in rule component. There are also established techniques for extracting rules from past observations, such as Inductive Logic Programming.

2.3.3 Model Interoperability

Figure provides an example illustration of the interactions among three "layered" PMESII models: social, infrastructure, and information models respectively from top to bottom.

The *infrastructure model* in the middle, represented as a Stabilization and Reconstruction Operations Model (SRO Model) (Robbins et al., 2005), captures a sequence of influences among variables, starting from the power supply at an electrical power substation. The generated power is fed into an industrial water plant, which produces water consumed by oil field work. An oil field produces crude oil to be refined by a refinery. Finally, refined fuel is used to generate power, which in turn is supplied to various power substations, thus forming a loop. It is especially difficult to reason with graphs containing such loops spanning many variables, as it creates an additional burden for discounting the variables' self-influence.

The *social model* at the top of the figure captures the influence of four variables, namely, power, drinking water, refined fuel, and sufficient food supply, on a variable representing the level of anger of the population in a town aligned with coalition forces. The dynamics of the social model are that short supply in any one of these three consumable products will increase the level of anger among the local population. In fact, if a terrorist organization became aware of the mid-layer SRO model sequence in the infrastructure then the power substation would assume heightened importance in the eyes of the terrorist strategists: attack on a substation would not only cripple other services along the sequence but would also drive the sentiment of the local population against the coalition. Note that the diamond box represents the expected mission utility in line with the level of anger. The utility (though difficult to quantify here) should go up when the anger level is down and vice versa.

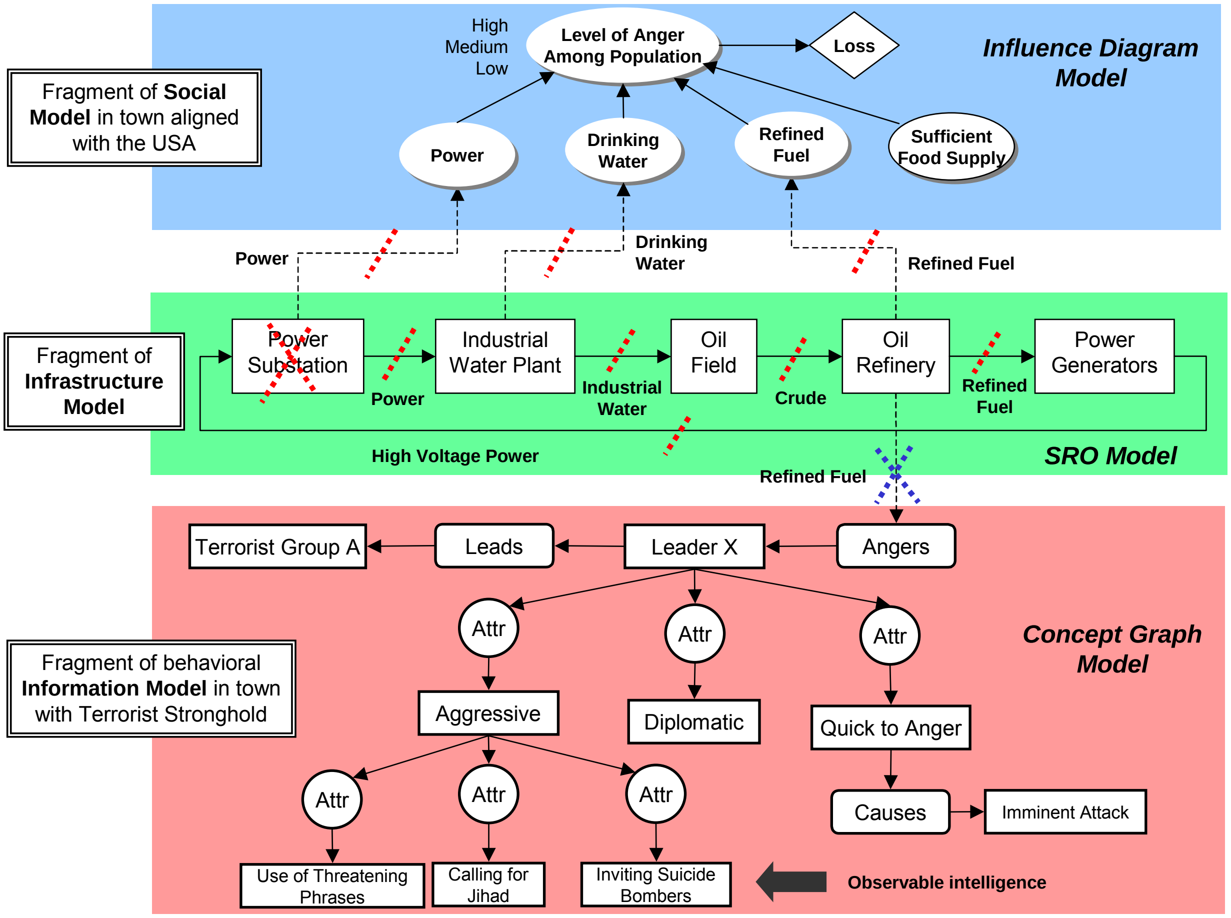


Figure 8: Interoperability of PMESII Models

The *behavioral information* model at the bottom of Figure 8 encodes a generic behavioral model of a terrorist leader in terms of a concept graph in which concepts are represented by rectangles, e.g., [Person: Leader X] and [Behavior: Aggressive], and conceptual relations are represented by circles and ovals, e.g., Leads, Causes. An analyst can query such a model to determine who the terrorist leader is, and the nature of the leader based on various observable intelligence. Such a leader X, who leads the terrorist group A, can possess different types of behavior attributes including aggressive, diplomatic, quick to anger, etc. If the leader is quick to anger and there are some stimuli to make the leader angry then an attack on friendly targets may be imminent. One such stimulus would be coalition forces stopping the supply of oil to the region, as indicated by the link between the SRO Model and the Concept Graph Model.

A key issue here, of course, is the interoperability among the models. Note that although an input/output connection has been made between the two variables *Oil Refinery* and *Refined Fuel* of the top two models, they are ontologically incompatible as defined earlier. However, they can be made compatible, by recognizing that the term *Oil* is synonymous to *Fuel*, and *Refined* and *Refinery* have a common base word. Another difficult compatibility problem is illustrated by the fact that there is no input for the variable *Sufficient Food Supply* in the social model, illustrating the interface incompatibility described in Section 2.1.1. One can envision, however, that this "sufficiency" concept could be automatically computed from the supply of food previously recorded in available databases, to bridge this last gap.

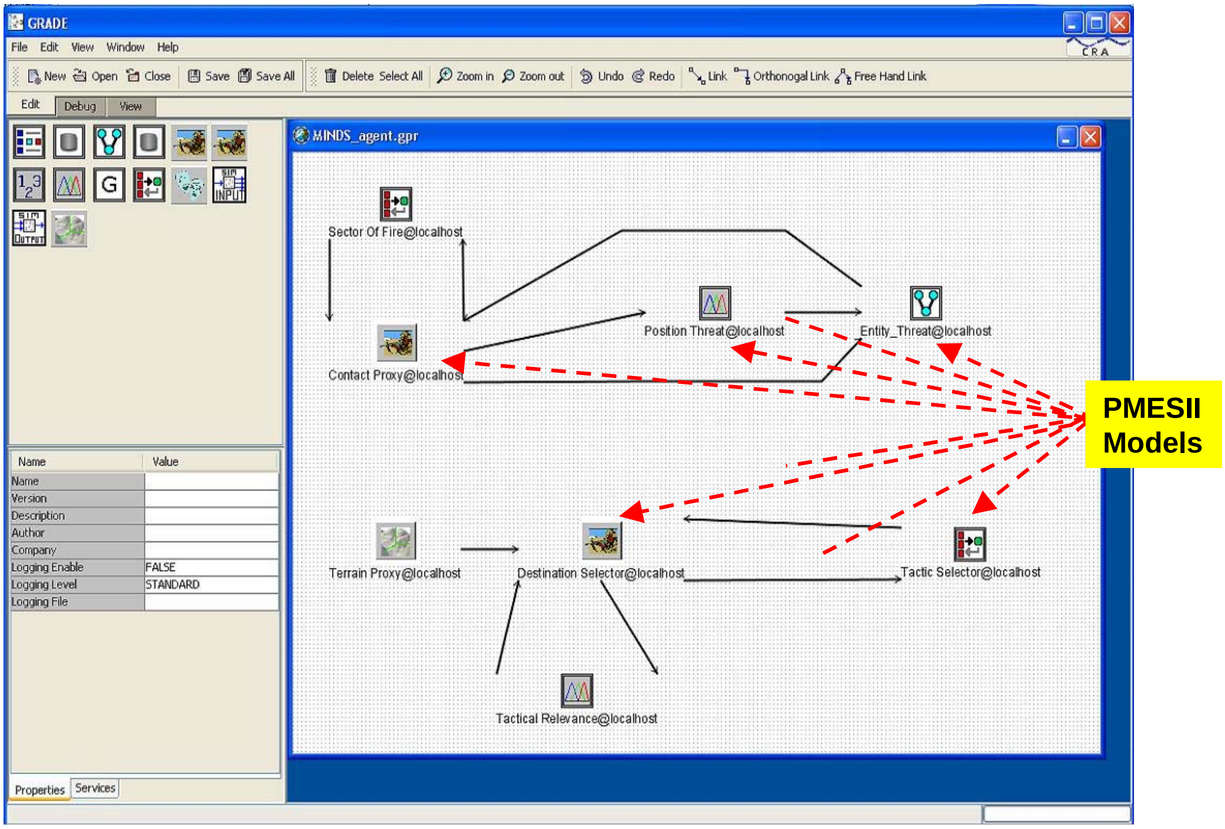
A number of recommendations for resolving specific model incompatibilities and functionality gaps are provided in Section 5. More general recommendations that can resolve more than one of these gaps simultaneously are presented in Section 4.

2.4 Implementation and Evaluation via GRADE

We implemented and evaluated CPEF using our in-house IDE GRADE (Harper et al., 2003). GRADE provides an XSLT translator for mapping inputs and outputs between models and a GUI for graphically integrating models by mapping their components to one another. GRADE was developed to enable rapid development of intelligent agents from existing components. For our purposes, a GRADE agent is a network of PMESII models and the components used to build an agent are PMESII models. Viewing a network of PMESII models as an agent is quite advantageous in situations where the system is required to run autonomously.

GRADE was developed to enable rapid development of intelligent agents without imposing the restrictive learning curve often associated with agent development. Ease of use has therefore been a top priority in our development of the toolkit. An agent in GRADE consists of the definition of a number of connected components describing the "mental model" that the human operator maintains throughout development. The mental model is instantiated as a feed-forward network defining both task interdependency and data flow through the model. GRADE is developed around this feed-forward network, providing the modeler with specific interfaces to add network components and fully define the functionality of those components.

GRADE has an intuitive graphical interface allowing the user to define the mental model topology and drill down into the individual components to fully define the underlying cognitive processing models. The main window frame of Figure provides the base interface for the agent model. The user can add components and define data flow between components of the agent model within this interface. On the left side of the interface is a palette from which the user can select new components to add to the network. This palette is populated at runtime by GRADE by adding all components to it that are installed in the component directory. The user adds new modules by selecting a particular node from the palette, and dragging the node into the drawing area. Once there, the user can specify the name of the component and configure it for runtime, e.g., define rules, create belief networks, etc. This will fully instantiate GRADE's data model for the node. Data links are specified between nodes by selecting the Connection Tool from the palette and drawing a line between the sending node and the receiving node.



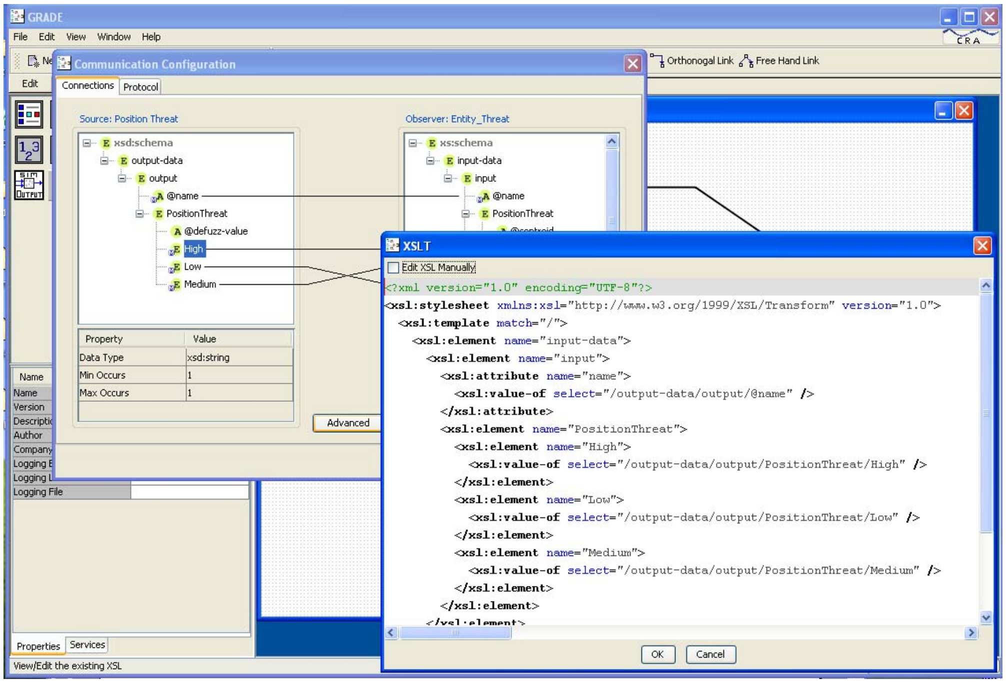
Name	Value
Name	
Version	
Description	
Author	
Company	
Logging Enable	FALSE
Logging Level	STANDARD
Logging File	

Figure 9: Building Networks of PMESII Models in GRADE IDE

The *AgentComponent* interface of GRADE defines the interaction between GRADE and each component editor that it uses. GRADE can be used to construct, edit and save any component that implements the *AgentComponent* interface. Once a component has been added to an agent, it can be configured by double-clicking its icon to access the component's own configuration GUI. GRADE displays the menubar and toolbar that are appropriate to that type of component. This allows the user full access to each of the agent's components, and also allows future third party components to be used within GRADE. GRADE's graph-based agent construction environment is enabled through the use of the JLoox toolkit (http://www.ilog.com/products/jloox/).

GRADE components communicate via XML messages. Our system design allows each component to employ different messaging formats defined by an XML schema, thereby allowing our current components, future components, and third-party components to communicate with one another. This framework provides an excellent integration point for the CPEF framework as integrating models can be seen as mapping their inputs and outputs so as to allow XML messages to be passed between them. In addition, models are typically stored in XML format.

To this end, we developed a graphical XSLT generator for GRADE. This tool allows agent developers to automatically define transformations between message formats through the use of a simple drag-and-drop interface. Figure shows the XSLT Generator Dialog. Each communications link acts to connect a source component, which produces output messages, with an observer component that receives those messages. The left side of this dialog shows the format of messages sent out by the source component. In this case, the source component is an instance of our Bayesian Network Component. The right side of the dialog shows the format of the messages that may be received by the observer component. In this case, the observer is an instance of our Logic Component.



Property	Value
Data Type	xsd:string
Min Occurs	1
Max Occurs	1

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:template match="/">
    <xsl:element name="input-data">
      <xsl:element name="input">
        <xsl:attribute name="name">
          <xsl:value-of select="/output-data/output/@name" />
        </xsl:attribute>
        <xsl:element name="PositionThreat">
          <xsl:element name="High">
            <xsl:value-of select="/output-data/output/PositionThreat/High" />
          </xsl:element>
          <xsl:element name="Low">
            <xsl:value-of select="/output-data/output/PositionThreat/Low" />
          </xsl:element>
          <xsl:element name="Medium">
            <xsl:value-of select="/output-data/output/PositionThreat/Medium" />
          </xsl:element>
        </xsl:element>
      </xsl:element>
```

Figure 10: GRADE XSLT Generator Dialog

The agent developer may select elements and attributes of the source message and drop them onto the observer message. This functionality is shown in Figure by the purple node-name attribute of the Bayesian Network Component. On the right panel, the node-name attribute is partially visible as it is being dropped onto an attribute of the Logic Component. We are continuing to work on this dialog so that a drag-and-drop action completed by the agent developer will result in GRADE automatically defining an XSLT instruction. This mechanism will allow the agent developer to define mappings between source and observer message formats by simply dragging and dropping between the left and right panels of the dialog. The new capabilities for easy drag-and-drop agent construction, easily accessible component configuration interfaces, and automated XSLT generation have greatly reduced the costs associated with model development.

GRADE includes three core components: fuzzy logic, Bayesian network, and rule base. The Fuzzy Logic component uses a proprietary Java-based fuzzy inferencing engine. It provides a full array of membership functions to define variable fuzzification as well as implication and connection methods for executing the fuzzy rules. The Bayesian Network component is based on Charles River Analytic's commercially available Bnet.Builder and Bnet.EngineKit tools. These provide the ability to design belief networks with a rich GUI and execute them quickly. Our component software wraps the functionality provided by the Bnet product. The Rule Base component is based on a proprietary forward chaining rule base system that we developed in Java for the component. We developed the graphical rule definition environment with a freeware library called Jazz, which has since been superseded by Piccolo (http://www.cs.umd.edu/hcil/piccolo/).

2.4.1 Relevance to CPE

The prime candidate program for transitioning CPEF is the Commander's Predictive Environment (CPE). The objective of the CPE program (Carozzoni, 2005) is to build a decision support environment for the JFC/JFACC and staff to better understand the mission space (past, present & future) and predict enemy intent, actions, and emerging threats in Joint Operations. CPEF supports this objective by allowing users to collaborate, extend existing models, and reason with networks of heterogeneous models, each of which may have particular strengths in modeling some specific PMESII domain. This provides for speculative "what-if" analysis and inference that helps predict enemy actions and responses in the overall social-political context of future military operations.

Our approach to the development of a methodological framework for building networks of integrated, interoperable, and adaptive PMESII models effectively responds to the CPE's need for enabling technologies that support:

- **Rapid PMESII Model Development:** CPEF facilitates skilled/non-skilled analysts rapidly building various types of graphical PMESII models to capture key relationships, dependencies, and vulnerabilities of adversary/self/neutrals.
- **Rapid PMESII Model Adaptation:** CPEF guides analysts in adapting existing PMESII models by identifying the input and output variables within a model and providing appropriate wrappers to expose those variables.
- **PMESII Model Integration/Interoperability:** Our approach integrates adapted PMESII models into networks of models and ensures their interoperability by addressing various types of gaps discussed earlier.
- **Maintenance of PMESII Models:** Our approach provides maintenance of adapted models by incorporating both static and temporal adaptation of existing PMESII models via machine-learning techniques and human refinements.

(Carozzoni, 2005) projects the CPE effort as a multi-disciplinary research. We describe below the disciplines that have been considered by Carozzoni and how CPEF fits into each:

- **Model Self/Adversaries/Neutrals:** The focus of CPEF is allowing users to build models and making them interoperable. Such models will capture the relationships among various friendly, adversarial, and neutral entities. Careful modeling and what-if analyses on these models will reveal strengths, capabilities, vulnerabilities and critical gaps.
- **Predict and Assess Probable/Multiple Courses of Action (COAs):** Some models will inevitably incorporate the notions of utility and thus whether a Red, Blue, or Grey COA is probable or not can be determined based on its expected utility.
- **Sensemaking Capability for Dynamic Battlefields:** This aspect is essentially about situation assessment and understanding of the mission at hand. Numerous models exist that provide the situation assessment functionality. CPEF allows for the integration of

multiple relevant models to expedite this process and also account for the fact that different modal formalisms will be applicable to different PMESII subdomains.

- **Knowledge Bases to Support Center of Gravity Analysis:** Knowledge bases within our framework are essentially networks of graphical models. Appropriate analyses on such interoperable models can generate the "center of gravity" in terms of network nodes that are highly sensitive in terms of system dynamics.
- **Operational Simulations for Mission Rehearsal:** The CPEF framework naturally allows multiple what-if analyses to carry out such operational simulations, with review of the results as provided by the constituent models.
- **Visualization:** The CPEF implementation platform GRADE has built-in capabilities for visualizing mission space, risks, uncertainty, and progress against objectives.

3. Project Task Performance and Results

In this section we present the tasks undertaken to fulfill the project objectives outlined in Section 1.2. This represents a record of the work performed during the CPEF research effort as well as a summary of the results that informed the recommendations in Sections 4, 5, 6, and 7. The project task schedule is shown in Figure . The remaining subsections are a breakdown of each task.

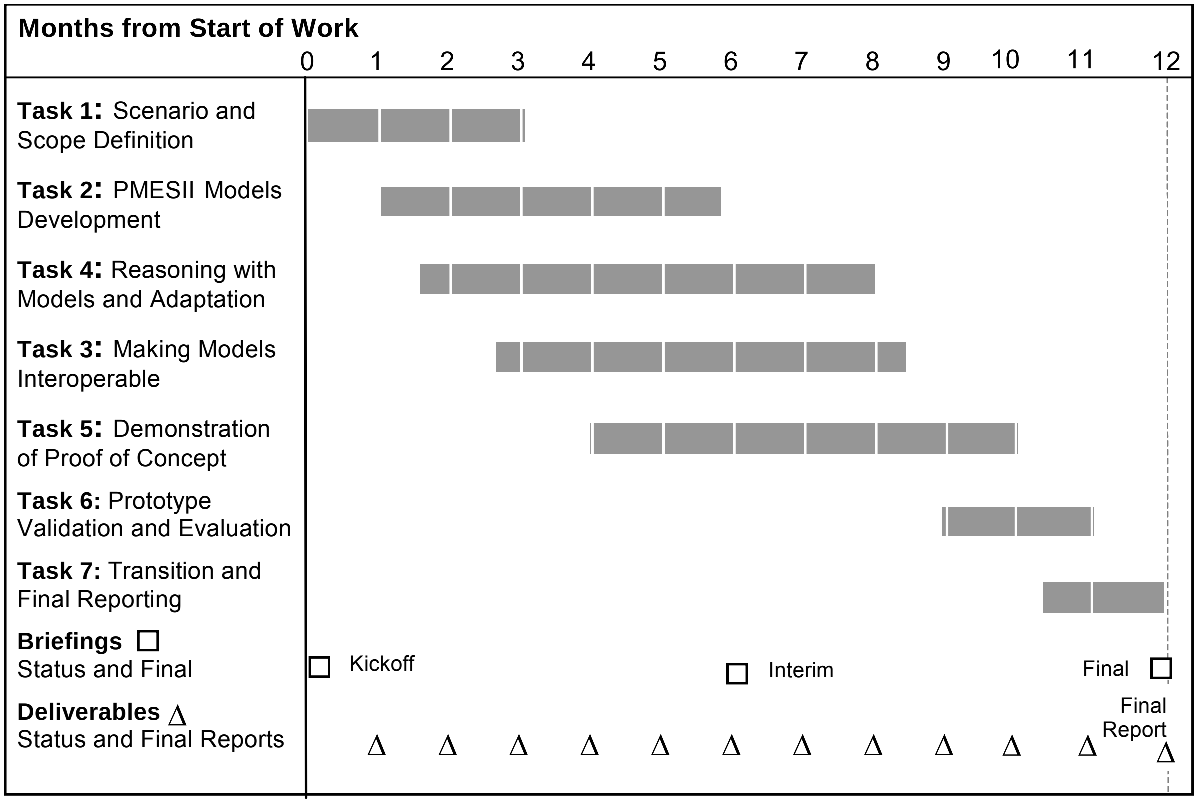


Figure 11: Project Task Schedule

3.1 Scenario and Scope Definition

As a result of discussions during the kickoff meeting and initial presentation, it was decided to constrain the focus of the effort around the application of developed technology to a concrete scenario. Candidate scenarios discussed were a North Korea/China scenario and the Iraq stability scenario. The Charles River team was asked to consider work already underway at AFRL/RI under the supervision of John Salerno building a system-dynamics model using the Ptolemy tool from the University of Berkeley. We received a sample core system-dynamics model, developed using Ptolemy, from AFRL/RI, and subsequently downloaded and installed Ptolemy from Berkeley. We then ran several demo examples that came with the software. Most of the examples were at a lower level (number crunching and signal processing) than we envisioned dealing with for the CPE. Ptolemy provides various modeling paradigms (FSM, SDF, CI are some relevant ones) for concurrent execution of interoperable models. The basic framework is actor oriented as opposed to object oriented. From this point of view, there were no required paradigm shifts associated with mixing and matching Ptolemy models with those from our agent-oriented GRADE system. Our initial experience with Ptolemy is reported in the following section.

We carried out a survey of existing PMESII models including the work of Capt. J.D. Robbins. We also found the ISSM site, (http://home.comcast.net/~dshartley3/TOOLBOX/issm.htm), among others. It seems that the owner of the site has integrated some PMESII models (in Excel spreadsheet form) and embedded them into DMSO's OOTW toolbox. The basis of his work is a 1999 CCRP publication entitled “Doing Windows: Non-Traditional Military Responses to Complex Emergencies” by Hayes and Sands . The book also mentions PMESII works from SAIC and EBR in the late 1990's using causal/influence network models. Much of this work subsequently contributed to DARPA's IBC program.

We narrowed the context down to a concrete scenario - the Iraq stability scenario. We also revised the Ptolemy model developed by AFRL/RI for use in CPEF implementation and evaluation.

To flesh out the domain of PMESII specification within the scenario and scope of CPEF we investigated the Effects-based Operation (EBO) ontologies created for AFRL/RI by ISX Corporation. They were captured using the Web Ontology Language (OWL). Adapting these files to Ptolemy or other formalisms has proven difficult. However, these ontologies did provide a great deal of context for the scope of CPEF. Our initial assessment was that we can develop an ontology actor in Ptolemy to facilitate interoperability among models.

3.2 PMESII Models Development

We were able to create a Ptolemy actor using the Ptolemy Java API, and then added it to the Ptolemy User Library to be used via drag-and-drop. As shown in Figure 12, the small actor that we created takes a Recruit as an input, e.g., the output Recruit values from the core system-dynamics model from AFRL, feeds this into a simple belief network model consisting of four variables: Recruit, Trainee, Active Duty, and Unemployed; and then outputs the posterior probability distribution of the Active Duty levels. The actor is then appended to AFRL’s model, as a demonstration of processing their outputs.

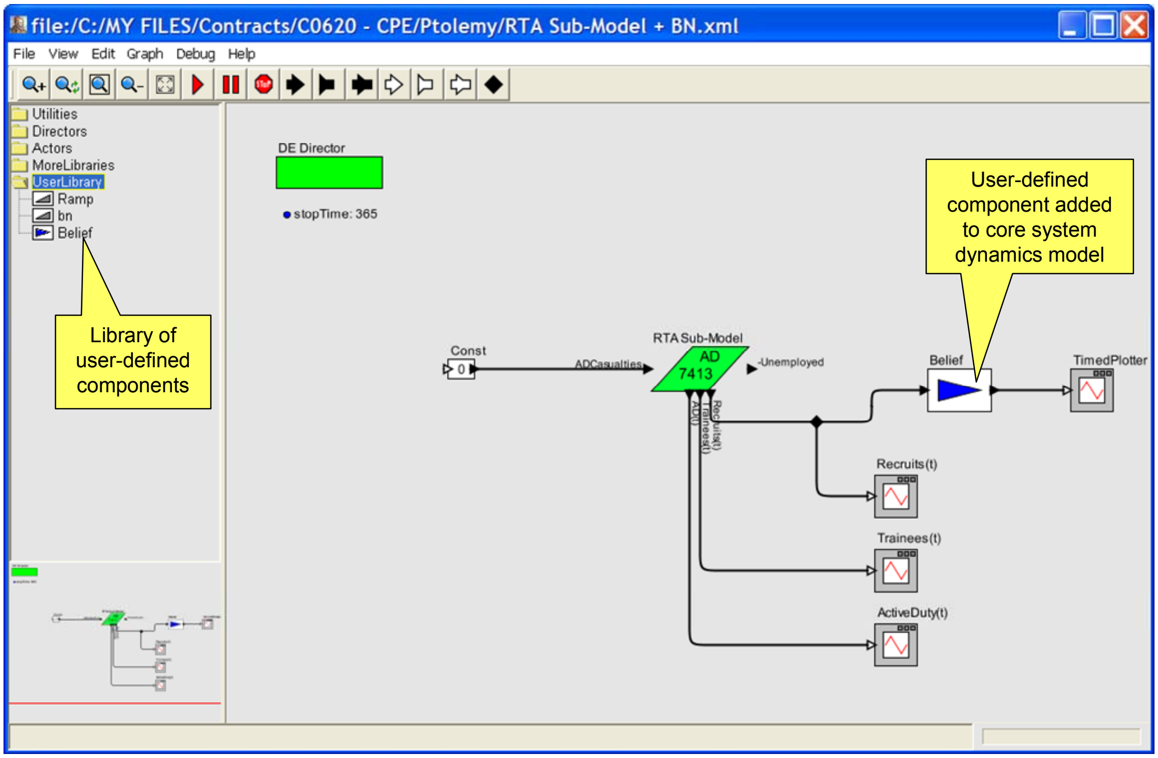


Figure 12: High-Level Core System Dynamics Model in Ptolemy

There are several conclusions that we can draw from this small effort, which will have an impact on the whole CPE program; AFRL's vision of building the core system-dynamics model in Ptolemy does not conflict with the proposed CPEF methodology; we can hook up our GRADE-based agents as Ptolemy actors, as we did in our simple scenario above, around the system-dynamics model; and therefore Ptolemy is a prime candidate for integration with GRADE. At this point Grade does not support SROM, Concept Graphs, and Concept Maps.

The clocked, time-synchronized processing activities in Ptolemy can enrich the entire interoperable modeling framework. We can easily envision various temporal PMESII models, e.g., Hidden Markov Model (HMM) and Dynamic Bayesian Network (DBN) that work in sync around the core system-dynamics model. The part that we liked most about Ptolemy is its drill-down visualization of embedded sub-models, which is an effective way to keep a large hierarchical type model manageable.

We have investigated how to incorporate ISX's ontology. An ontology-based translation process can be the job of a GRADE actor with knowledge of various ontologies.

We continued working with AFRL to revise their Iraq stability model. We integrated their Health Model, shown in Figure 2 and built with Ptolemy, with our Bayesian Network (BN) based Political Influence Model shown in Figure , developed using GRADE.

The Political Influence Bayesian network model (PIM) integrates daily health data from the AFRL/RI Health Model (RLHM) and calculates from that data: economic, civil unrest, service interruption, and morale factors. The variables for day 0 are “health0”, “econ0”, “unrest0”, “services0”, and “morale0” respectively. The daily morale factor will be used to predict a political influence coefficient whose day 0 variable is “mutable0”. The variables for day 1 to 5 are named accordingly. This model need not be a completely representative calculation, but served as a proof of concept mechanism for model integration. The PIM model developed is a dynamic Bayesian network and provides data for six days of activity.

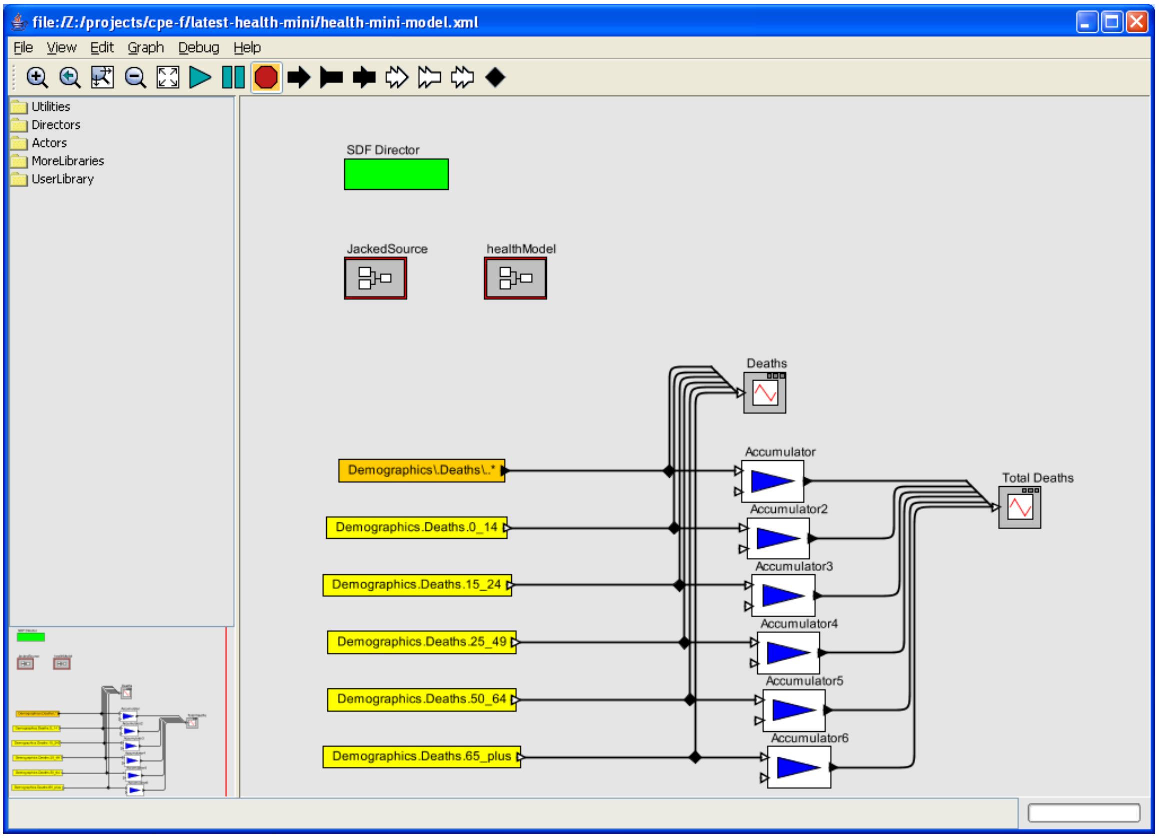


Figure 2: AFRL/RI Health Model Version 1

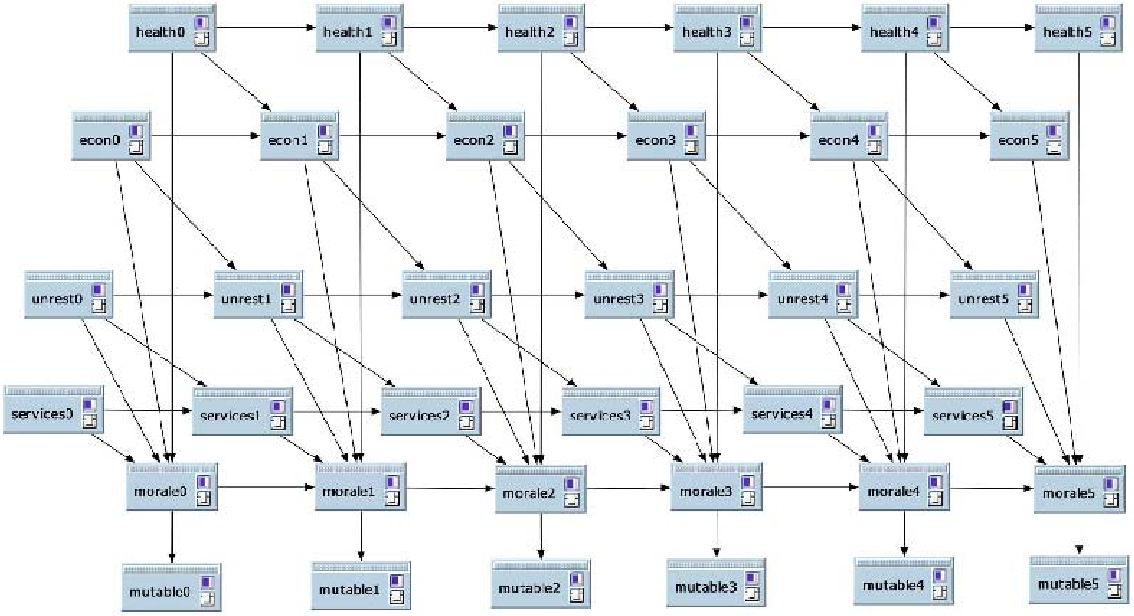


Figure 14: Political Influence BN Developed Using GRADE

Revisions of the AFRL Iraq stability model were ongoing. The next iteration of the health model is shown in Figure 3. The interactions between the variables of the health model shown in Figure 3 and the BNet model shown within in GRADE in Figure 4, are precisely defined in a Condition Probability Table. For example, variable "econ1" depends on "health0" and "econ0." Because there are two possible states (good and bad) for both inputs there are four values to be defined for "econ1's" good state and bad state. The values are specified as percentages. The following are the values used in this model for "econ1":

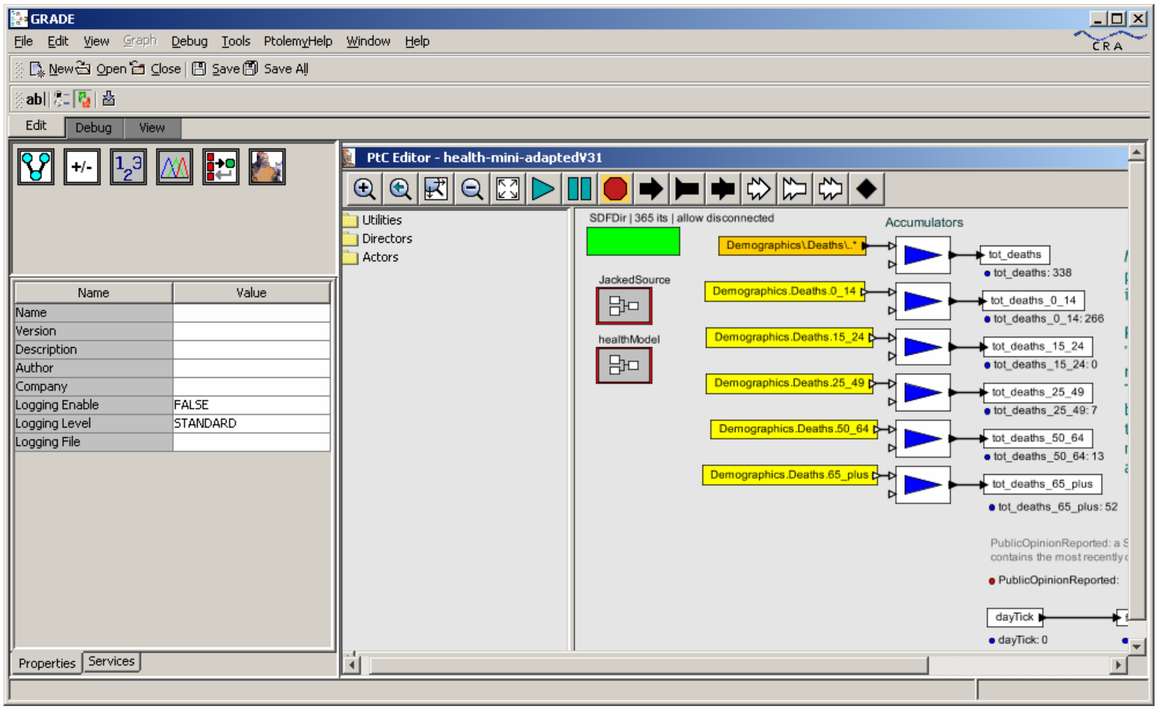
econ0: good; health0: good => econ1: 100% good

econ0: good; health0: bad => econ1: 95% good

econ0: bad; health0: good => econ1: 20% good

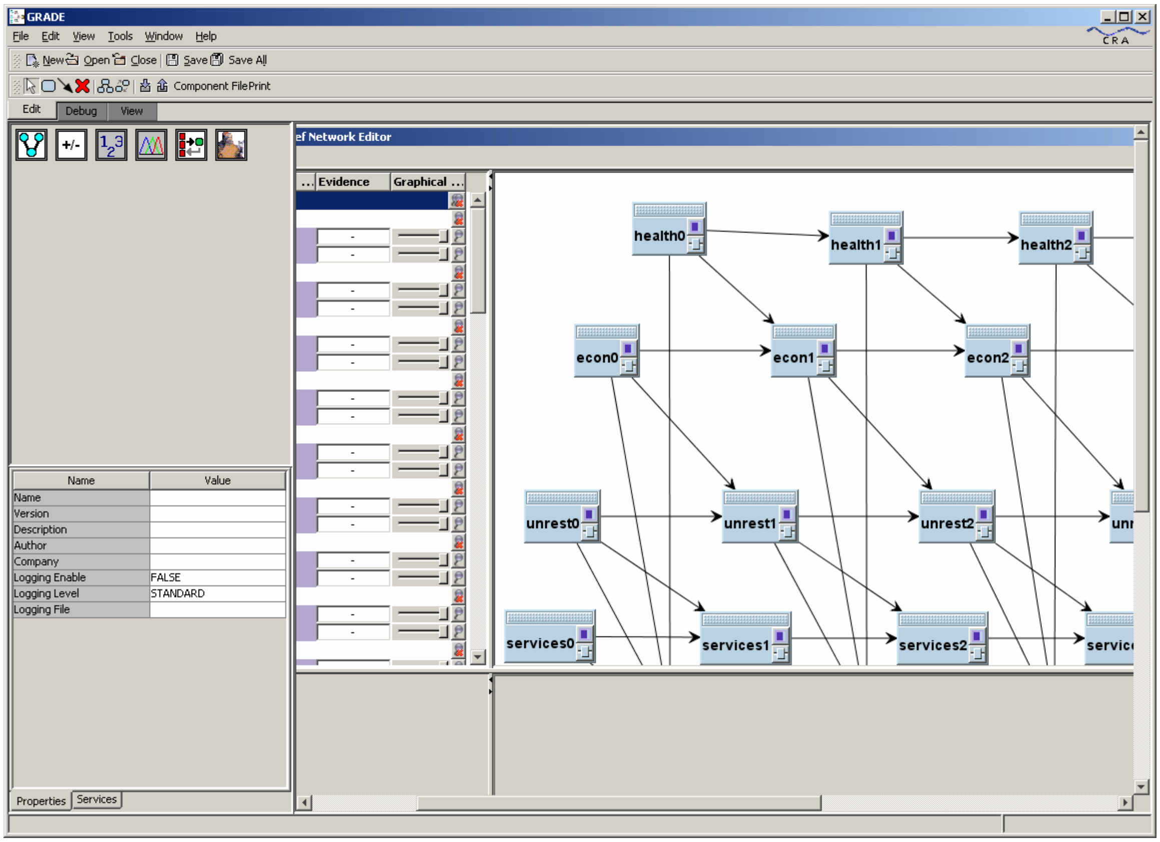
econ0: bad; health0: bad => econ1: 0% good

The variables for day 1 to 5 are named similarly to day 0. This model need not be a completely representative calculation, but is a proof of concept mechanism for model integration. The PIM model developed is a dynamic Bayesian network and provides data for six days of activity.



Name	Value
Name	
Version	
Description	
Author	
Company	
Logging Enable	FALSE
Logging Level	STANDARD
Logging File	

Figure 3: AFRL Health Model Version 2



Name	Value
Name	
Version	
Description	
Author	
Company	
Logging Enable	FALSE
Logging Level	STANDARD
Logging File	

Figure 4: The Political Influence within GRADE

3.3 Reasoning with Models and Adaptation

ISX Corporation made the EBO core ontologies available to CPE contractors as part of the CPE program. This set of ontologies captures a representation of nodal (in the EBO sense of the term) interdependencies and vulnerabilities, along with the actions that exploit those vulnerabilities, and also how changes are induced directly via action flow. The fundamental relationship patterns in the node ontology are replicated at all levels of specificity. We studied the ontologies in detail to determine how they can be used within a framework for ontological inference. Such inference can be used not only for model integration but also for reasoning with networks of heterogeneous models. Our initial assessment is that we can develop an ontology actor in GRADE to facilitate interoperability among models.

We reviewed the Effects-based Operation (EBO) ontologies from ISX Corporation for applicability to our effort. The ontologies are captured using the Web Ontology Language (OWL). Ptolemy uses a different dialect of XML and adapting these two formats is one of the issues we identified in Task 4 that needs to be addressed.

Stanford University has a tool called Protégé for working with OWL files. We downloaded Protégé and the OWLViz plug-in to display ISX's CPE ontologies. Figure shows a visualization of the EBO ontology using Protégé.

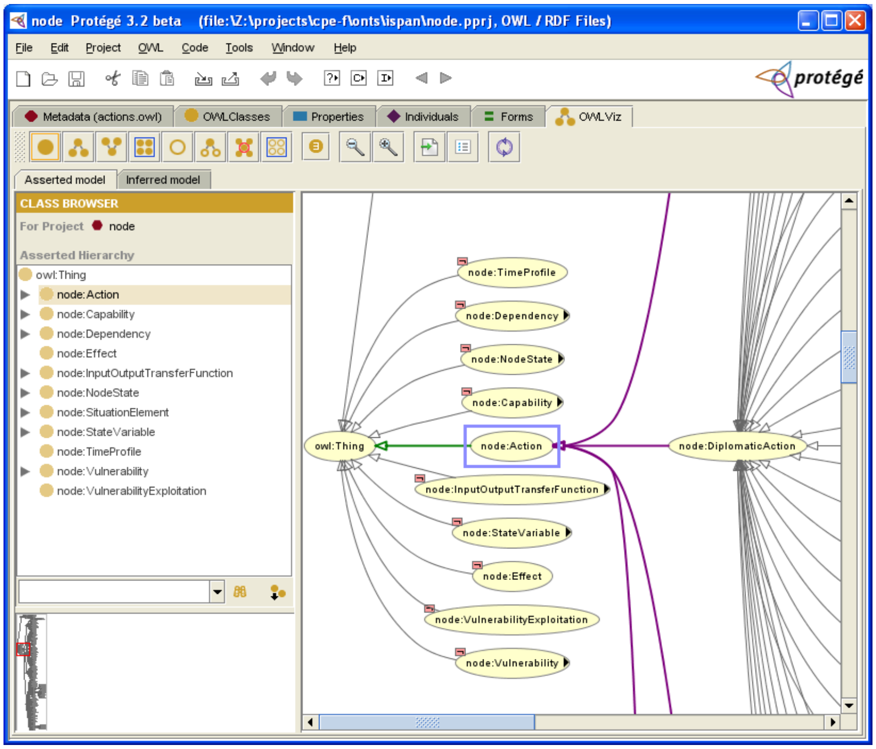


Figure 17: Visualizing ISX's Model with Protégé and the OWLViz Plug-in

As was discussed in the proposal, adaptation between models may have to occur at several levels: interface, ontological, formalism, and sub-domain incompatibility. Clearly not all models will be incompatible at all levels. However, in order to use the ISX model with a Ptolemy model it needs to be integrated at the interface level. One possible approach considered is a Java adapter component based on the toolset available from the Source Forge Jena Project (jena.sourceforge.org). It provides a Java API for processing OWL ontologies. An alternative would be to use XSLT. We analyzed the applicability of these options.

This task included a review of formal ontology specification languages. OWL-DL is based on a *SHIQ* Description Logic and is in fact equivalent to the *SHOIN*(D_n) DL. *SHIQ* and *SHOIN*(D_n) are common DL descriptions for which features an attributive DL language allows—one letter per feature set, e.g., *I* is the set of "inverse properties". OWL-DL benefits from years of DL research and has well defined semantics. Its formal properties are well understood. Known reasoning algorithms exist, and highly optimized systems have been implemented around OWL-DL.

An OWL-DL ontology is known in DL terminology as a Knowledge Base. It has two parts: the Terminological Box (TBox) and the Assertion Box (ABox). The TBox contains axioms about classes - OWL axioms such as subclass, equivalent class, or disjointness axioms. The ABox contains assertions about "individuals" - OWL facts such as type, property-value, equality, or inequality assertions. Reasoners operate on Knowledge Bases.

We reviewed a number of enabling technologies for using ontologies to reason about models. Pellet is an open-source reasoner that works with OWL-DL. It accepts the SPARQL protocol and RDF query language (SPARQL) and provides a Java API to make it easy to integrate into custom systems. According to the literature, Pellet is a complete and capable reasoner with very good performance, extensive middleware, and a number of unique features.

We designed a functional process where the XSLT Generator drives a Pellet reasoner. It was to be written in Java using the Jena toolkit API (http://jena.sourceforge.net/). The generator was to create an XSLT translation which 1) accepts the output of the Ptolemy model, and 2) converts it to an acceptable input for the Bayesian Network. The XSLT Generator was to configure the Pellet reasoner to analyze the two models, send SPARQL requests to the Generator, read the results, generate XSLT, and save the XSLT generated.

An Xalan translator was reviewed to process the XSLT generated from the Pelet reasoner. Xalan would read the first model's XML output and convert it to XML to be read by the second model. We planed that the CPE Framework will add XML adapters to any models which do not generate or accept XML natively.

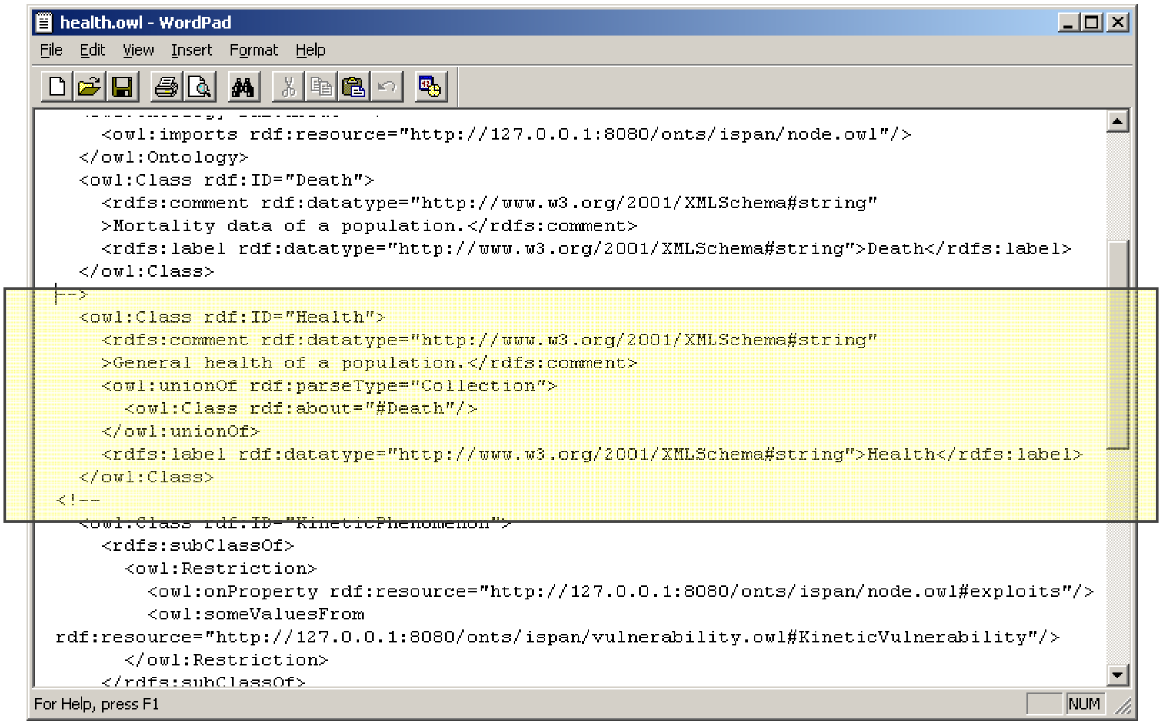
SPARQL, mentioned previously, is a query language. From the W3C (http://www.w3.org/) draft:

> "The SPARQL query language consists of the syntax and semantics for asking and answering queries against RDF graphs. SPARQL contains capabilities for querying by triple patterns, conjunctions, disjunctions, and optional patterns."

We researched how to connect two models with respect both to their data *and* to their semantics. The key difficulty in connecting the models is understanding what the data they process represents. At this point we encountered and documented the utility of metadata within the CPEF framework and how CPEF might use varying levels of metadata specification since we can not rely on users or other frameworks to provide this specification. In all designs, our approach was geared towards supporting users, but still requiring them in the process of model integration.

We revised the larger CPEF framework to incorporate the GRADE/Sample application, Ptolemy and BNet. In the new design, the analyst will first select the models he wishes to integrate. The models will then be read into GRADE which parses the XML descriptions of each. Next the output and input identifiers are parsed by the CPE Framework. It checks the identifiers for the substring matches and for the longest alphabetic string contained in each to be used as a token. The input tokens and output tokens are then checked against an ontology/taxonomy description to determine if they are related using the Jena toolkit API written for Java (http://jena.sourceforge.net/). The CPE Framework stores this information for use below. We also envisioned the use of WordNet as a key taxonomy resource, but other taxonomies/ontologies were also to be investigated. We show in Figure 5 a simple OWL syntax example we used to relate health to death.

When the analyst connects the two models on-screen using GRADE, a Communication Configuration pane is displayed identifying the available outputs and inputs. Selecting an output checks the available inputs for related identifiers using the information stored in the previous step and highlights them for the analyst. This provides a visual filter to aide in selecting the correct input.



```
    <owl:imports rdf:resource="http://127.0.0.1:8080/onts/ispan/node.owl"/>
  </owl:Ontology>
  <owl:Class rdf:ID="Death">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Mortality data of a population.</rdfs:comment>
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Death</rdfs:label>
  </owl:Class>
-->
  <owl:Class rdf:ID="Health">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >General health of a population.</rdfs:comment>
    <owl:unionOf rdf:parseType="Collection">
      <owl:Class rdf:about="#Death"/>
    </owl:unionOf>
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Health</rdfs:label>
  </owl:Class>
<!--
  <owl:Class rdf:ID="KineticPhenomenon">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="http://127.0.0.1:8080/onts/ispan/node.owl#exploits"/>
        <owl:someValuesFrom
rdf:resource="http://127.0.0.1:8080/onts/ispan/vulnerability.owl#KineticVulnerability"/>
      </owl:Restriction>
    </rdfs:subClassOf>
```

Figure 5: OWL Description of Health and Death Relation

We further investigated the implications of reasoning within a network of integrated, heterogeneous models. Recommendations based on our findings are provided in Section 7.

3.4 Making Models Interoperable

One of the first ideas considered in this effort was the use of Concept Maps to define the relationships between parts of different models. The Institute for Human and Machine Cognition (IHMC) has an application, CmapsTools, which provides a convenient mechanism to capture Concept Maps. We downloaded their product and analyzed its usefulness for the purposes of CPEF. One attractive feature is that it provides exporting of Concept Maps to XML. Our hope was that this would provide a means to define the interoperation of disparate models. Figure 19 shows an example.

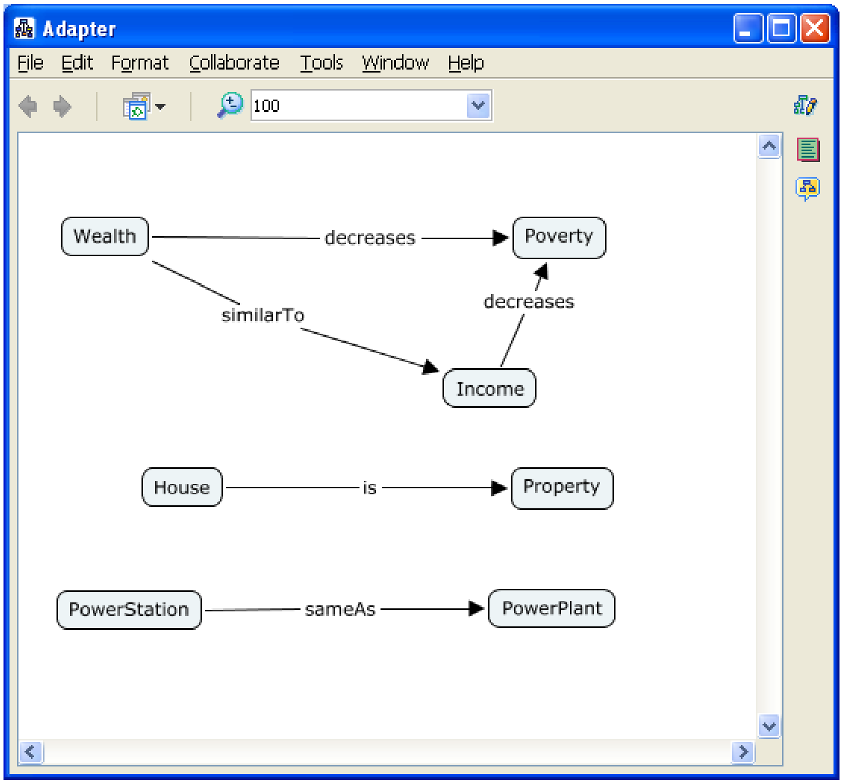


Figure 19: Using Concept Maps to Relate Synonyms from Two PMESII Models

Concept Maps are able to define many types of relationships such as: synonym (A is-the-same-as B), near-synonym (A is-similar-to B), antonym (A is-the-opposite-of B), proportional influence (A increases B), etc. The power of the tool is the flexibility of what it can describe. Unfortunately the tool does not provide any limits either, so there is a risk that non-domain specific relationships could add noise to the definitions if care is not taken. This is one of the aspects of this approach that ultimately led to us not incorporated it within GRADE or using it for further development of CPEF. However, it remains a candidate that could be analyzed for use in a full-scale CPEF system.

We were able to leverage other in-house works-in-progress to integrate Ptolemy models into Bayesian belief networks. Two network fragments were constructed. The first describes the dynamics of the number of police officers and the second describes the dynamics of the number of criminals. The fragments are linked: the number of police officers influences the number of criminals, and both influence the number of unemployed.

Both fragments are dynamic Bayesian networks. They are designed in three layers. The first layer represents the fraction of the population in different categories at the previous point in time. Each such variable is denoted by the name of the category and **Minus,** e.g. **IncarceratedMinus**. The second layer consists of transient variables that represent the fraction of the population changing categories. The third layer contains variables representing the fraction of the population in each category at the current point in time.

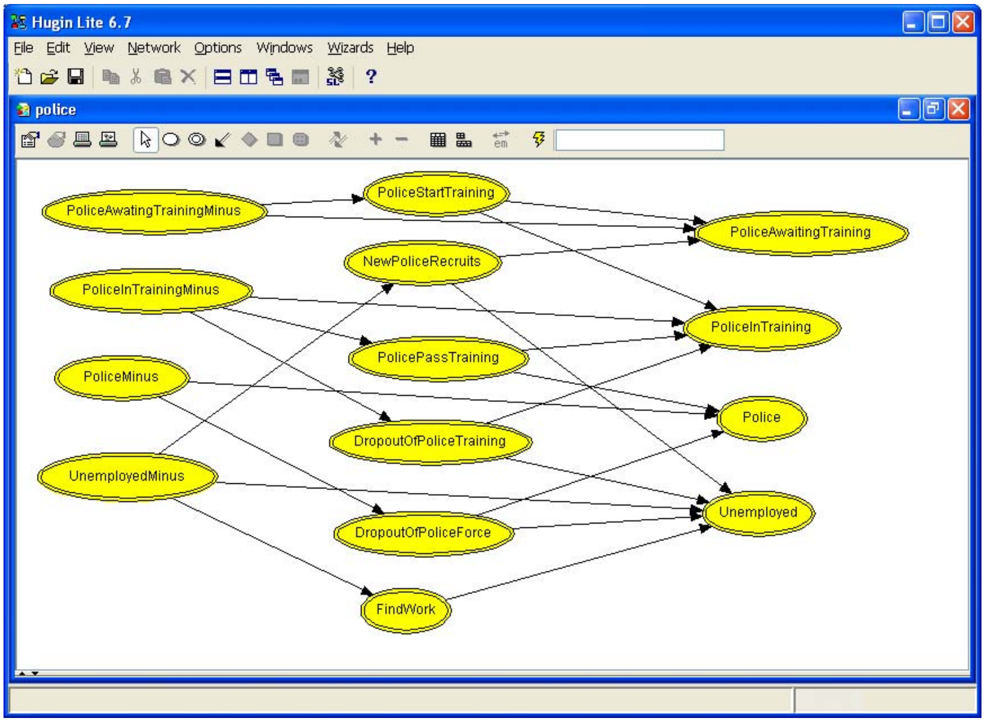


Figure 6: The Dynamics of the Number of Police Officers

Each variable in the second layer is defined to be a percentage of a category variable at the previous point in time. For example, the fraction of people released from jail is a percentage of the number of people incarcerated at the previous point in time. It is defined by the formula:

Released = N (0.1 IncarceratedMinus, 0.0004).

This indicates that **Released** is a Gaussian distributed random variable whose mean is **0.1 IncarceratedMinus** and whose variance is 0.0004.

These intermediate variables are useful because they characterize exactly how the fraction of people in each category evolves. In addition, they serve to tie the rate of evolution of different categories. For example, people who drop out of the police force are subtracted from the number of police and added to the unemployed.

There is an exception to this pattern. Sometimes the level of one category influences how many people will enter or leave another category. For example, the amount of police influences what fraction of criminals becomes incarcerated. Modeling this properly would require non-linearity, because we would take a fraction of the previous level of criminals and multiply it by the level of police. But the framework only allows linear Gaussian distributions. So we have modeled the situation by saying that the level of police has an additive effect on the number of people becoming incarcerated. That is:

NewlyIncarcerated = N (0.1 CriminalsMinus + 0.2 PoliceMinus, 0.0004).

Variables in the third level integrate the changes in the second level. The fraction of the population in a category begins with the fraction at the previous time point, and adds or subtracts the fraction of people entering or leaving the category. For example, we have:

Incarcerated = IncarceratedMinus + NewlyIncarcerated – Released.

There is no noise in these variables; they are simply summations.

Sometimes the people leaving one category may be split up into more than one new category. For example, some people released from jail may become criminals again, while others go back to joining the general population. We have modeled this by adding only 0.6 times **Released** to **Criminals**.

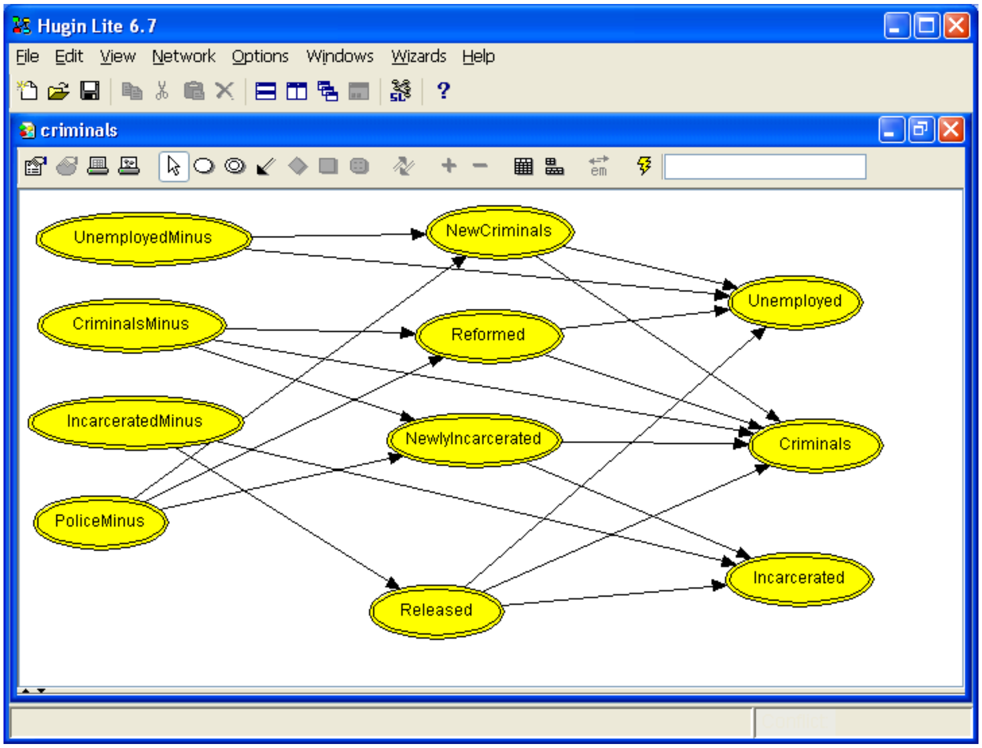


Figure 21: The Dynamics of the Number of Criminals

The variables in the first layer are inputs into the model. No probability model is defined for them. In reality, they will depend on the levels at time *t*-2 by the same mechanism shown here. This will go back to time 0. To complete the model, a probability distribution needs to be specified over the levels at time 0. This is an ordinary Bayesian network.

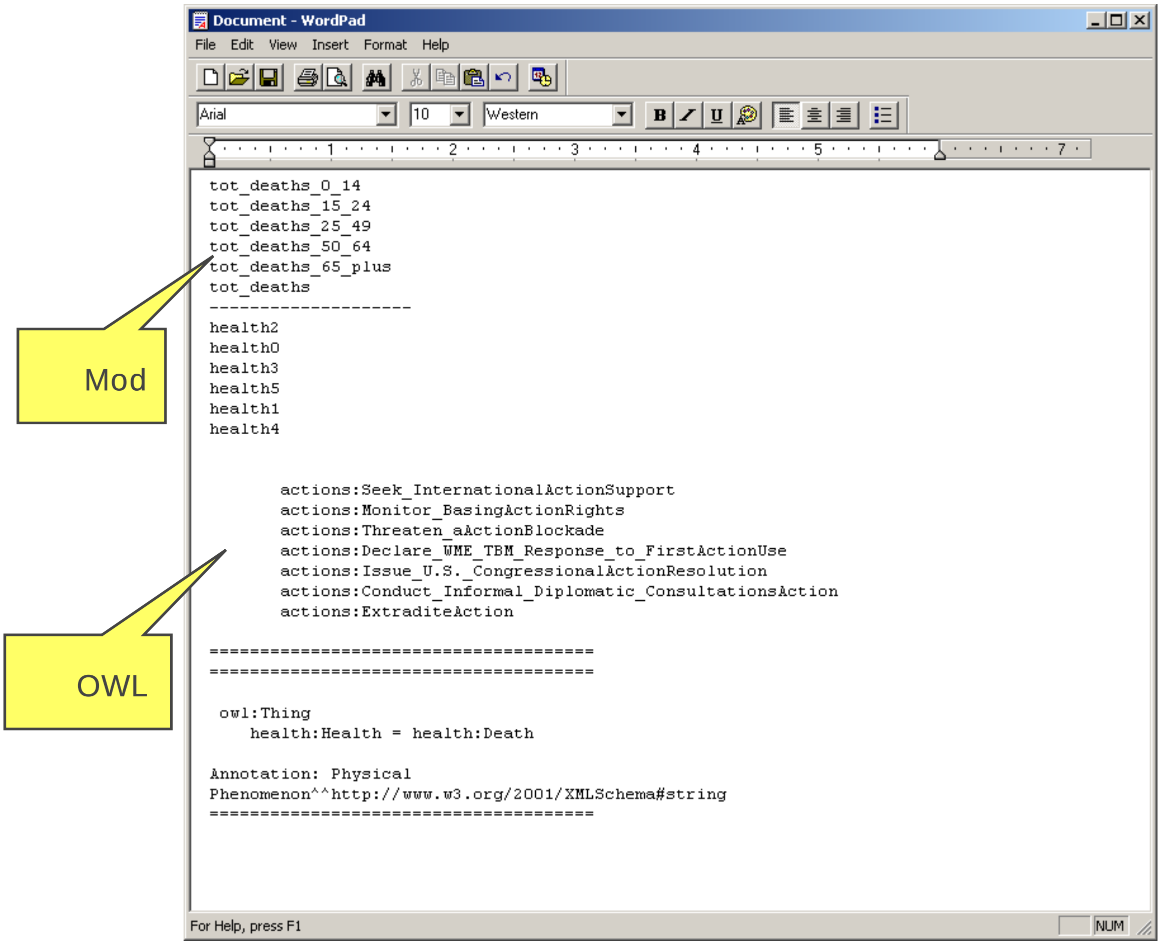
Note that there are probability models for **Unemployed** in both fragments. In a full model that combines the different fragments, there will only be one **Unemployed** variable that is outside all the fragments. The model for this variable will be the summation of the different models described in the different fragments.

Further research proved that OWL could provide a more comprehensive and relevant syntax to define the relationships needed for our project. OWL was therefore investigated further and ConceptMaps were left for later investigation, given time. Additionally, there is an ontology library that has been built in the public domain provided by the DAML Project. We decided to look further into how that resource could be used as a starting point for our reasoning and interoperability efforts.

We extended our CPEF design for the purposes of PMESII model interoperability. The CPE Framework operates as part of the GRADE Communication Configuration pane. Once the models are loaded into GRADE, model integration is performed in six steps:

1) The models are identified and the Configuration pane is instantiated,
2) The models' XML is parsed to generate a list of identifiers,
3) These identifiers are further parsed to generate a list of tokens,
4) These tokens are used, in output/input pairs to generate a list of potential relation matches,
5) Once the Analyst selects an output identifier from the list of possible outputs, CPEF highlights the inputs which matched it, and
6) After selecting the desired input, GRADE generates an XSLT translation for the connection.

Figure shows identifiers of the Ptolemy, B Net, and EBO/OWL models which have been parsed. JDOM was used to parse the first two models; direct Pellet calls were used to parse the OWL. DOM node names are parsed and node attributes with the value "name". To parse the OWL, a Pellet Reasoner instance is created for the OWL model and its `getClass()` method is invoked with a URI, e.g., http://127.0.0.1:8080/onts/ispan/health.owl#Health.

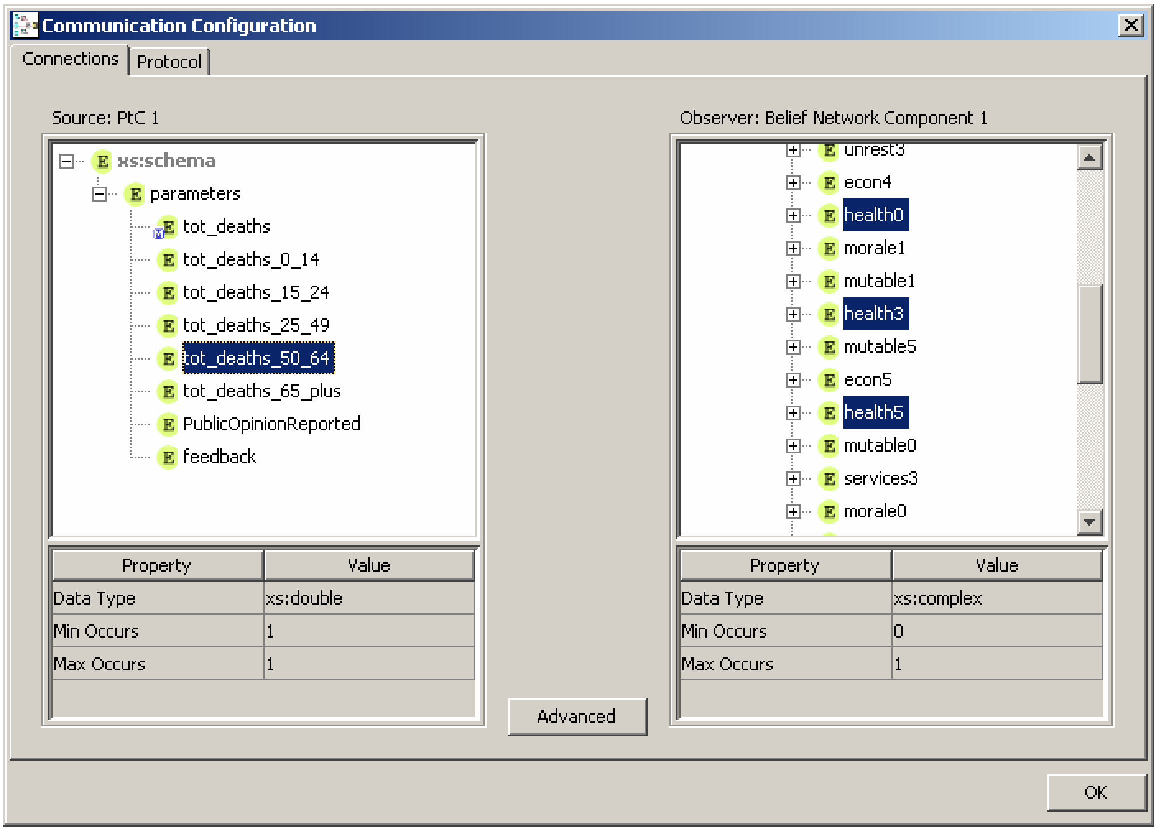


```
tot_deaths_0_14
tot_deaths_15_24
tot_deaths_25_49
tot_deaths_50_64
tot_deaths_65_plus
tot_deaths
--------------------
health2
health0
health3
health5
health1
health4


       actions:Seek_InternationalActionSupport
       actions:Monitor_BasingActionRights
       actions:Threaten_aActionBlockade
       actions:Declare_WME_TBM_Response_to_FirstActionUse
       actions:Issue_U.S._CongressionalActionResolution
       actions:Conduct_Informal_Diplomatic_ConsultationsAction
       actions:ExtraditeAction

======================================
======================================

 owl:Thing
    health:Health = health:Death

Annotation: Physical
Phenomenon^^http://www.w3.org/2001/XMLSchema#string
======================================
```

Figure 22: XML Model and OWL Ontology Parsing

3.5 Demonstration of Proof of Concept

We initiated work on the proof of concept demonstration to target the scenarios developed and models built by AFRL. A demo was coded to demonstrate the integration of the health model developed with John Salerno and the economic model developed in-house. GRADE was used to read in the two models, CPEF identified related inputs utilizing Semantic Web technologies, and the user was presented with recommendations for model component mapping. Figure shows the initial prototype used for proof of concept demonstration.



Property	Value
Data Type	xs:double
Min Occurs	1
Max Occurs	1

Property	Value
Data Type	xs:complex
Min Occurs	0
Max Occurs	1

Figure 23: Using Semantic Web Technology to Highlight Related Inputs

Figure 4 shows the final limited-scope prototype. It included further support for semantic analysis including an interface to WordNet and Roget's Thesaurus. It also has a clear and extensible API for constructing custom dictionaries. For example, one could include a dictionary of common PMESII terms and their relations to one another. This would help in integrating models when comparing the semantic similarities of their structure, text labels, and any other text meta-data specification. It is important to notes that this semantic analysis goes much deeper than simply see if words are synonyms. It includes a number of word relations including antonyms, hypernyms, hyponyms, etc. Thus, a structural analysis of two models and how they may integrate can be performed via a semantic analysis by looking at the text labels of their nodes.

3.6 Prototype Validation and Evaluation

The actual metrics and evaluation of the limited-scope CPEF prototype is described in Section 3.6.3. We defined a number of metrics for future evaluation along the lines of a user study as described in Section 3.6.1. The approaches, designs, and recommendations that have been researched and defined during this contract can only be completely evaluated with non-experimental PMESII models, experienced analysts as test subjects, and a full-scope prototype implementation. The metrics described here can therefore serve as recommendations for future evaluation of a full-scope CPEF and / or related systems.

For an evaluation baseline or ground truth we recommend the use of the current state of model development. Models acquired during this project have been created in both Ptolemy and BNet. Our ground truth will therefore consist of how analysts currently use these tools to create models, deduce how models should interact, manually connect models, and then reason with integrated models. Evaluation should then be based on a comparison of how well users perform these tasks with and without the supports of CPEF. The models used for evaluation can be based on those created during Task 3, AFRL Health Model (RLHM), and any other relevant PMESII models.

3.6.1 Evaluation Metrics

Metrics for evaluating CPEF fall into two categories: 1) those that evaluate its algorithms accuracy, and 2) a user study that evaluates its efficacy and utility. Some simple tests that fall into the first category are described in Section 3.6.3. Metrics that can be used in a future user study to quantify ease of model integration with and without CPEF supports are:

- GOMS analysis: This is a simple yet effective measurement of the number of mouse clicks, keyboard strokes, and other interface interactions necessary to perform model integration. The goal would be for CPEF to require less interaction than current manual model integration.
- How quickly a user joins models: This can be measured by users integrating similar models with and without CPEF supports and measuring the time differential in each case.
- How accurately a user joins models: Accuracy can be subjective, thus we must try to choose model structures that reduce ambiguity but do not trivialize the problem of model integration. Accuracy measurements will then depend on how well causal and logical structures of individual models are retained for reasoning in integrated models. Accuracy measurements will be compared for models that have been integrated with and without CPEF supports.

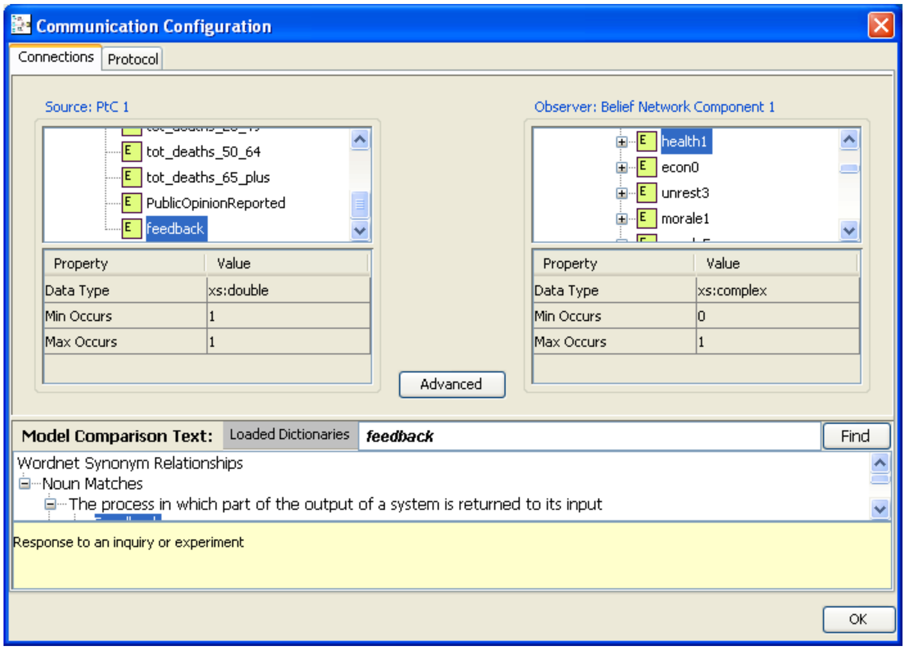
3.6.2 Evaluation Experiment

We have designed a user study for future evaluation of the CPEF system. Users will be presented with a set of models within the GRADE interface and asked to connect them for interoperability. In some sessions, the GRADE interface will include recommendations made by the CPEF system. In other sessions it will not. A GOMS analysis will be performed for each session, i.e., the number of interface interactions, e.g., mouse clicks and keyboard strokes, the user must make to integrate models will be counted. The duration of each session will also be measured. The GOMS analysis of each session will be compared to determine whether the CPEF supports require less, more, or the same number of interface interactions as manually connecting PMESII models in GRADE. The desirable outcome would be for the CPEF supports to reduce the number of interactions necessary, and the time required to connect models. This would indicate an increase in the speed and productivity of the user.

A second experiment could aim to measure the cognitive support provided by CPEF. Models will be pre-constructed to contain certain system dynamics, and to interact with each other in a particular manner when integrated accurately. The models will contain only single integration points, i.e., there will be only one way to accurately connect the models for inter-operation. Users will then be asked to connect and reason with the models, with and without CPEF supports. They will also be asked to record their observations. This will be a within-subjects experiment, i.e., what sets of models are given to users with and without CPEF supports will be interleaved to rule out learning and bias. Measurements will then be made according to whether users accurately connected models with more, less, or the same accuracy when using CPEF supports vs. without. Observations of users will also be analyzed to see if they were able to make insights about the known system dynamics with or without CPEF supports.

3.6.3 Evaluation of the Limited-scope CPEF prototype

The actual evaluation of CPEF centered on assessing the accuracy of our integration of semantic web technologies to support users integrating PMESII models. Figure shows the CPEF prototype displaying a semantic analysis of the PMESII model node “feedback”. The semantic analysis is performed using WordNet and Roget’s Thesaurus but can incorporate other ontologies and lexical databases as described in Section 6. The semantic analysis of the “feedback” node is shown in the bottom frame of Figure . The evaluation consisted of comparing the prototype’s semantic analysis output to the output of a manual analysis via the WordNet online interface shown in Figure . Indeed, one can see that the analysis presented in the lower frame of our prototype in Figure is identical to the output of WordNet shown in Figure for the word “feedback.”



Property	Value
Data Type	xs:double
Min Occurs	1
Max Occurs	1

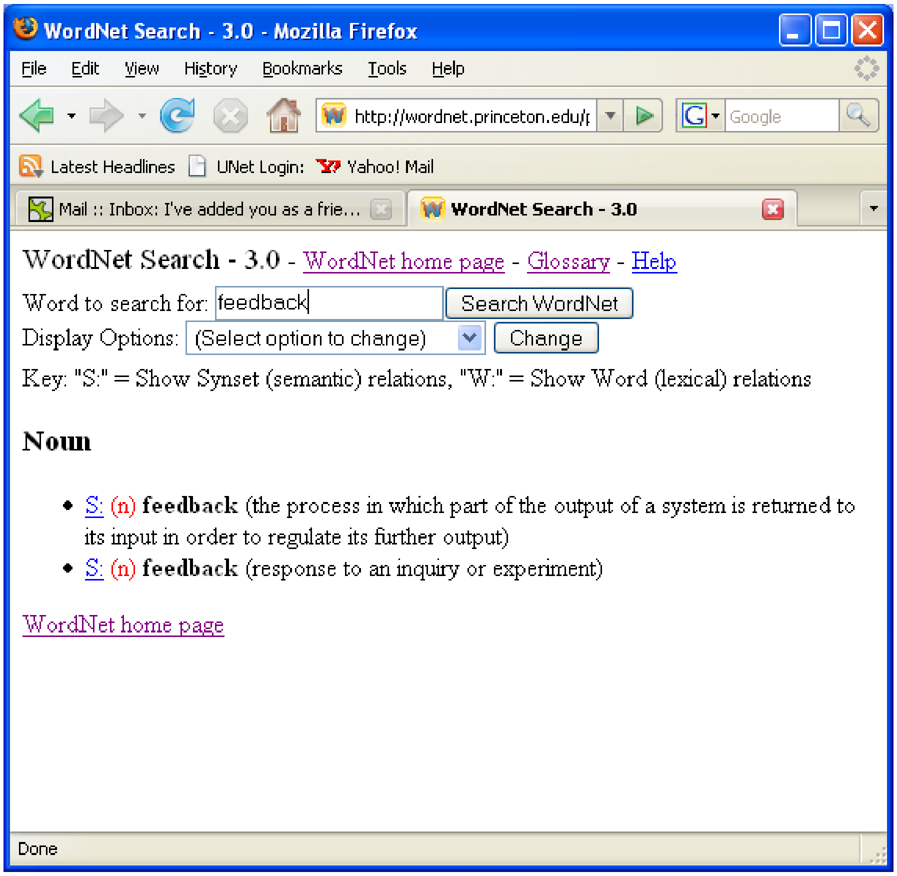
Property	Value
Data Type	xs:complex
Min Occurs	0
Max Occurs	1

Model Comparison Text: feedback

Wordnet Synonym Relationships
Noun Matches
The process in which part of the output of a system is returned to its input

Response to an inquiry or experiment

Figure 24: CPEF Prototype Displaying Semantic Analysis of PMESII Model



WordNet Search - 3.0 - WordNet home page - Glossary - Help

Word to search for: feedback

Display Options: (Select option to change)

Key: "S:" = Show Synset (semantic) relations, "W:" = Show Word (lexical) relations

Noun

- S: (n) **feedback** (the process in which part of the output of a system is returned to its input in order to regulate its further output)
- S: (n) **feedback** (response to an inquiry or experiment)

WordNet home page

Figure 25: WordNet Web Interface Showing Semantic Analysis of "Feedback"

3.7 Transition and Final Reporting

We have compiled the observations and results of the CPEF effort and document them within this report. This includes an outline of our performance during this effort as outlined in this section, recommendations for the construction, adaptation, and integration of PMESII models, a system design for a full-scope CPEF prototype, and a methodology for reasoning with networks of heterogeneous yet integrated models. We have also investigated use of the methodology and algorithms developed under this effort in our commercial products GRADE and BNet.

4. System-wide Recommendations for PMESII Model Integration

The recommendations in this section represent approaches that can be used to address multiple interoperability gaps as described in Section 0. Recommendations that are specific to each interoperability gap are presented in Section 5. Sections 4 and 5 are highly related and cross-reference each other; however, we have separated them here for clarity of scope.

4.1 Mixed Initiative Learning

CPEF and related systems can learn mappings between PMESII model types from user interaction. The advantage of this approach is that it does not require elaborate logics or algorithms; it simply stores the history of user actions and attempts to predict future actions based on that history. Consider a user adding the metadata tag "entails" to a directed edge between nodes A and B on a rule-based PMESII model. CPEF may store this information and suggest the metadata tag "entails" in the future when the user is again constructing a rules-based PMESII model. Such suggestions are not limited to metadata entry, but can also include mappings between models of different formalisms. In fact, such an approach can learn from every interface manipulation that the user produces, during any phase of model construction or use. There is obviously a threshold for such mixed initiative approaches, and the user should be able to view, evaluate, edit, accept, or reject CPEF suggestions. The high level idea however is that CPEF can simply observe user interactions, learn from repeated use cases, and make informed suggestions for meta-data tagging and model integration. These suggestions would improve over time as the system learns from repetitive use cases.

4.2 Collaborative Learning / Filtering

A similar approach to mixed initiative learning is collaborative filtering. The primary difference between the two is that with collaborative filtering, interactions from multiple users are aggregated to predict the future actions of individual users and produce better suggestions for model integration. Because of differences in analysts' conceptions of PMESII models, it is not clear how affective this approach would be.

4.3 Ontologies: XML Files, XML Schemas, Metadata, and Inference

We can not expect or depend on the specification of PMESII ontologies for model integration. Nor do we wish to constrain users by requiring such a specification. However, we can put structures in place to facilitate both the specification of metadata, and the automatic derivation of ontologies where possible, to support model integration through ontological inference. The motivation for using ontologies is discussed in Section 4.3.1 while issues and complications of their use are described in Section 4.3.1.

As part of this discussion we must define what constitutes an ontology. At the highest level (and its philosophical roots), an ontology is a categorization of entities and the relationships between them (Gruber, 1993). An XML file or schema that defines a PMESII model or set of models can therefore be considered an ontology. This is discussed further in Sections 4.3.4 and 4.3.5. Users with the expertise and impetus can also specify formal ontologies as described in Section 4.3.5. Sections 4.3.7, 4.3.8, and 4.3.9 present methodologies for generating ontologies using different levels of user specification, including none at all.

CPEF is designed to use ontologies in support of PMESII model integration and reasoning with networks of heterogeneous models. Section 4.3.10 describes the traditional use of ontologies while Section 4.3.11 describes how CPEF departs from this convention. Finally, Section 4.3.12 provides an overview of the inference techniques that ontologies make possible, which are the central motivation for using ontologies in CPEF.

4.3.1 What can Ontologies Encode?

An ontology can ultimately encode all information necessary to connect two or more PMESII models and bridge all incompatibilities and functionality gaps outlined in Section 2.1. More specifically, ontologies can encode the following elements of a PMESII network:

- Nodes and the edges between them, thus the structure of the network
- Meta-data descriptions of the functions of nodes, edges, networks and sub-networks
- Data type restrictions on nodes and edges
- Information on model/network data flow, computation, and aggregation of values
- A taxonomy of formalisms and instructions on how to translate from one to another
- Explicit instructions for data translation from one network to another
- Explicit instructions on how to structurally connect models (e.g. one sub-network in one model to one node in another)

4.3.2 Why Use Ontologies?

A users' motivations for using ontologies and the functionality they provide are:

- Easier model construction:
    - Models can be re-instantiated rather than constructed from scratch

    - Users can construct models with predefined model components and templates instead of specifying all necessary information from scratch
    - Better graphical user interface support - ontologies and models templates and components can be shown as a pallet of model widgets which can be dragged into place to form a model
    - All of the data specified for a model as summarized in Section 4.3.1 can be re-used and shared
- Easier model sharing
    - Models can be translated using XSLT for storage and sharing
    - Models can be translated from one format to another
- Easier Model integration
    - Models can be modified as the resolution, purpose, and scope of modeling changes, e.g., users can append model A by adding a sub-network already defined in model B
    - To examine higher level system dynamics, two or more models can be integrated more easily
- Little to no change in current practice:
    - To motivate the use of ontologies we try and make them unobtrusive so that no extra work is required. Users can thus include as little or as much meta-information as desired – although it will always be the case that more specification yields better results in model integration.
    - Little to no specification can be used (see Section 4.3.4 for using XML files as ontologies)
    - Ontologies can be learned from a database of models
    - Ontologies can be evolved by extending existing PMESII models when building new ones
    - Users can explicitly specify points of integration and how models should be mapped

System-level motivations for using ontologies is their ability to support integration of PMESII models, and also their support for reasoning with heterogeneous networks of PMESII models to determine high level system dynamics. There are two primary aspects of ontologies that support CPEF:

- **Ontological Inference:** Ontologies support several forms of inference (see Section 4.3.12) that can be used for performing model integration and reasoning with heterogeneous networks of models. Even partial specifications of ontologies, including XML files, can be used in reasoning algorithms to determine the best mapping between elements, inputs, and outputs of two or more PMESII models for integration
- **Knowledge Representation:** with ontologies, users can directly specify how elements of their PMESII model should be mapped to other models or other formalisms, e.g., Ptolemy, BNets, Influence Diagrams, Concept Graphs, etc. They can also provide an unlimited amount of meta-data that describes the function of nodes, edges, sub-networks, and networks.

In the case where only node and edge labels are provided for partial specification is available, we can reason about the following items for model integration:

- The number of incoming and outgoing edges to an element
- The child / parent relationships of nodes
- What paths can be traveled from one element to another
- The semantic meaning of the label for a node or edge

Reasoning about most of these items can be performed as graph traversals, however some, such as a semantic analysis of the label on a node or edge can not. Ontological inference is also a superset of other inference mechanisms such as graph comparison. Ontological inference can potentially be used to address every model interoperability gap described in Section 2.1:

- **Interface Incompatibility:** Ontologies can specify how outputs of certain model nodes or elements should be mapped to the inputs of another. Typically, XSLT will be used for the purpose of translation and XML schemas will describe the data types of the elements, inputs, and outputs. In this scenario, the XML schema itself is the ontology.
- **Ontological Incompatibility:** Ontologies can provide metadata describing the function of their components. This metadata can then be used to translate from one ontological framework to another.
- **Formalism Incompatibility:** Users can explicitly specify how a model should be implemented across formalisms with an ontology. Further, a taxonomy can be derived to provide defaults. For instance, it can be determined that when integrating a

Bayesian network with a neural network that a "true" output of a Bayesian node should map to a "fire" input of a neural network node.

- **Subdomain Gaps:** Ontologies can be extended to cover broad domains. If two models cover very different domains, then an ontology can be derived that covers both and namespaces used to disambiguate any redundancy. The relationships between these domains and their models can then be specified to facilitate model building, adaptation, and integration.

The rule of thumb is that the more metadata specification in an ontology, the stronger its faculty of inference. If ontologies are fully specified and used in model creation as described in Section 4.3.8, then most interoperability gaps can be easily solved. This is because users can explicitly specify how one model should be mapped to another. A system like CPEF would simply load this information and execute the ontology as a set of instructions during model integration.

4.3.3 Ontology Issues and Complications

4.3.3.1 Who Specifies the Ontology?

Ontologies must be specified, therefore we must first determine how this is to be accomplished. Within CPEF we do not want to constrain users by requiring them to specify ontologies, nor do we expect users to obey such constraints. In addition, different users will have different conceptions and ontology specifications, sometimes for the same PMESII model. Standards can be developed, however they must evolve as the battlespace and PMESII models change. Simple issues of concurrent model specification can invalidate such standards, i.e., 2 analysts specifying the same model at the same time but using different ontology elements for specification.

We recommend that solutions *should not* require ontology specification, but support it where possible. Inference can be performed on data structures such as XML files of varying levels of specification which we can consider ontologies as described in Section 4.3.4. More meta-data means more sophisticated approaches can be leveraged; however, we can not assume or require that users specify meta-data. CPEF puts ontology management in the hands of users rather than programmers, but allows them to enter as much or as little metadata as they are able and willing. Some of its methods utilize ontological inference given only the state in PMESII models and / or their XML schemas. We recommend a number of techniques that utilize ontological inference but account for variable amounts of metadata. These include relation mapping presented in Section 5.2.3 and model node aggregation presented in Section 5.2.4. Ultimately, however, it would be desirable to have complete ontologies. Sections 4.1 and 4.3.8 suggest two methods for deriving such ontologies without requiring users to author a monolithic, static, and therefore brittle ontology such as Cyc.

4.3.3.2 One Master Ontology or Several Associated Ontologies?

It is anticipated that any attempt to author a complete PMESII ontology that encompasses every possible model will invariably fail. This is primarily because it is impossible to accurately predict every potential structure and/or use for a PMESII model. Systems such as Cyc, which tried to codify all "common sense" knowledge, typically result in extremely complex and brittle sets of rules that are incomplete and do not cover every possible situation. To avoid these issues, we *do not* recommend that CPEF use one master ontology. Tim Berners-Lee discussing the State of the Semantic Web in IEEE Intelligent Systems speaks of using "folksonomies," loosely federated small local ontologies, as opposed to the "Ontology of Everything" approach of Cyc, citing the need for solutions that scale well. We recommend similar approaches as described in Sections 4.3.7, 4.3.8, and 4.3.9.

4.3.3.3 Limits of Inference and Understanding

Efforts in natural language processing (NPL) to date have been unsuccessful in using ontologies for understanding human language. It is important to note that CPEF does not use ontologies for this purpose. CPEF uses ontologies to perform inference on PMESII models to support their integration and interoperation. It can use XML files and schemas and the state information in PMESII models for ontological inference. Simply put, if one can represent models for file persistence, e.g. XML, then one can represent them with ontologies. If one can represent models with ontologies, then one can use the set of ontological inference techniques to manipulate and reason about them.

4.3.4 XML File as Ontology

CPEF can use an XML file as an ontology to perform inference in support of model integration. Information that can be used in comparing two elements of XML files for matching and subsequent integration includes:

- The number of outgoing edges
- The labels on nodes and edges
- The depth
- Parent, sibling, and other node relationships
- The number and values of attributes on nodes

- The data type of the values specified for nodes

For instance, nodes α of model A and μ of model B may be matched because they both have 5 child nodes and a "grandparent" node with similar labels. This information would be clearly represented in the XML files that stored these models on disc. The more metadata that is specified, the more powerful the inference that can be performed. To further our example, if both nodes had a "description" attribute and text that was semantically similar (this can be determined through semantic analysis as described in Section 5.2.2), then we could match these nodes with an even greater degree of certainty. All of this can be accomplished through the minimal specification available in the XML file persistence of a PMESII model.

4.3.5 XML Schema as Ontology

XML schemas are XML files that specify constraints on the structure, elements, and data types of elements to be used in a set of XML files. For instance, a schema may specify that any conforming XML files must have an element "economy" which takes on a "complex value" with a value cardinality of three. The three values could be "good", "moderate", and "bad." XML Schemas can be considered to specify the set of XML documents that conform to their constraints. An XML file that conforms to an XML schema can be called an "instance" of that schema.

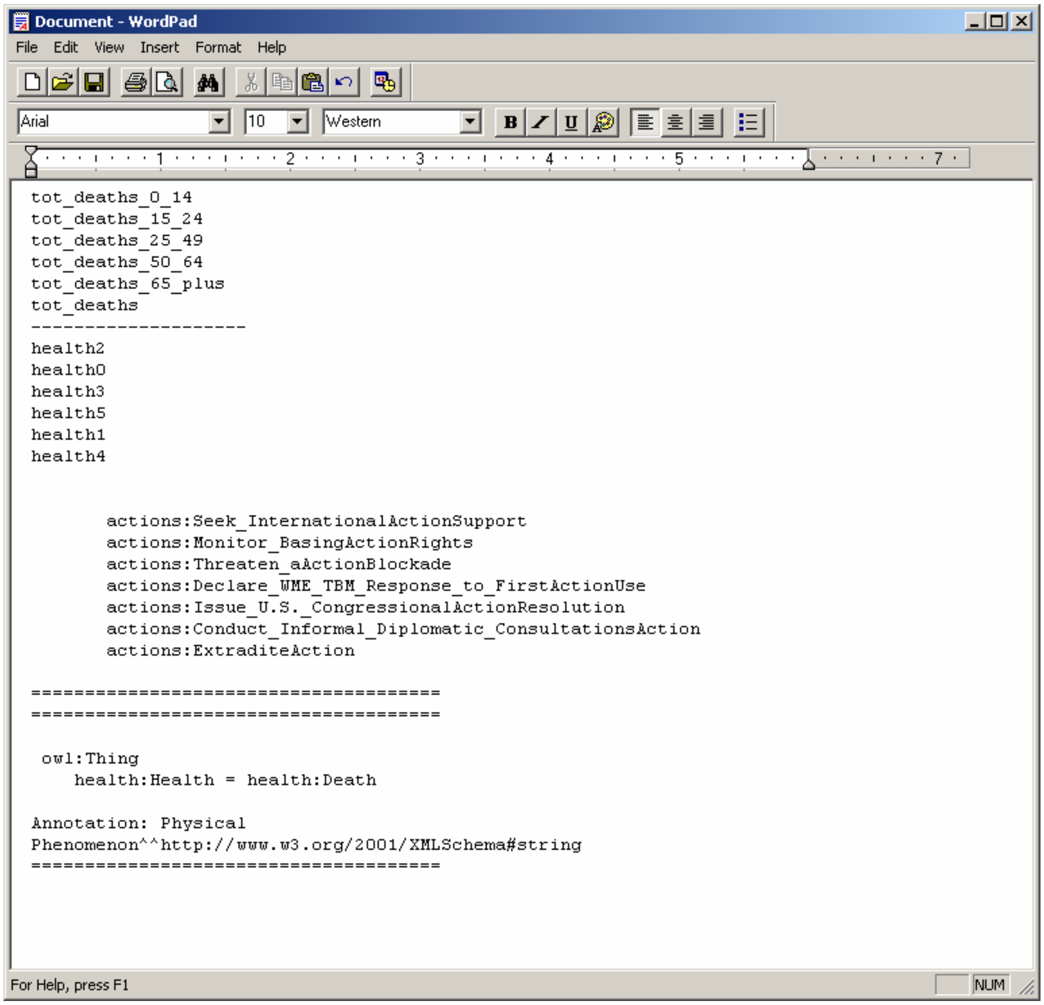
In many ways, XML schemas provide greater support for ontological inference by providing more detailed specification. Information that can be used in comparing elements of XML Schemas for matching and subsequent integration includes:

- The data types of values that elements can take on
- The cardinality of values that elements can take on
- Explicit specifications of how data types from one element can be mapped to the data types of another
- Metadata specifying

4.3.6 Formal Ontology Languages

Users who have the impetus and expertise can specify PMESII models in formal Description Logic syntax such as RDF and OWL. OWL-DL is based on a SHIQ Description Logic and is in fact equivalent to the SHOIN(Dn) DL. OWL-DL benefits from years of DL research and has well defined semantics. An OWL-DL ontology is known in DL terminology as a Knowledge Base. It has two parts: the Terminological Box (TBox) and the Assertion Box (ABox). The TBox contains axioms about classes - OWL axioms such as subclass, equivalent class, or disjointness axioms. The ABox contains assertions about “individuals” - OWL facts such as type, property-value, equality, or inequality assertions. There are several implementations of reasoners that perform inference on Knowledge Bases.

Figure shows identifiers of the Ptolemy, BNet, and EBO/OWL models which have been parsed using JDOM for XML and Pellet for OWL. DOM node names are parsed and node attributes with the value “name”. To parse the OWL, a Pellet Reasoner instance is created for the OWL model and its getClass() method invoked with the URI “http://127.0.0.1:8080/onts/ispan/health.owl#Health”.



```
tot_deaths_0_14
tot_deaths_15_24
tot_deaths_25_49
tot_deaths_50_64
tot_deaths_65_plus
tot_deaths
--------------------
health2
health0
health3
health5
health1
health4


       actions:Seek_InternationalActionSupport
       actions:Monitor_BasingActionRights
       actions:Threaten_aActionBlockade
       actions:Declare_WME_TBM_Response_to_FirstActionUse
       actions:Issue_U.S._CongressionalActionResolution
       actions:Conduct_Informal_Diplomatic_ConsultationsAction
       actions:ExtraditeAction

======================================
======================================

 owl:Thing
    health:Health = health:Death

Annotation: Physical
Phenomenon^^http://www.w3.org/2001/XMLSchema#string
======================================
```

Figure 26: Parsing Model and Ontology Specification

4.3.7 Automated Tagging, XML Schema generation, and Ontology Learning

Dr. Fernando Gomez has used syntactic and semantic analysis for automated annotation of text documents (http://www.cs.ucf.edu/~gomez/). This and similar approaches can be used to derive ontologies from the labels, descriptions, and structures of PMESII models. More specifically, ontologies and XML schemas can be derived from XML files used for PMESII model file persistence. Further, XML schemas can be derived for a set of XML files. Given a set of PMESII model XML files, an algorithm for deriving an XML schema would operate in the following steps:

1. Cluster the XML files into classes based on similarity in structure and content
2. Start an empty XML schema for each class
3. For each class and for each document in that class:
    a. Examine the data types of each element in the document
    b. If the schema doesn't currently represent that element, add such an element to the schema
    c. If the schema doesn't currently represent the data type of the element, and the data type does not conflict with any existing data types specified for that element, then add that data type specification to the element in the schema
    d. If the document contains an element that can not be represented in the schema given the constraints of prior steps, then create a new schema and class and add the document to it

This process can be executed on any collection of XML documents. The least amount of information in an XML schema and / or ontology generated from such an algorithm would be the data types, possible values, and structures of particular classes of PMESII models. In some cases, labels and descriptive text for some models may be similar. This text could be used to further specify metadata within resulting ontologies.

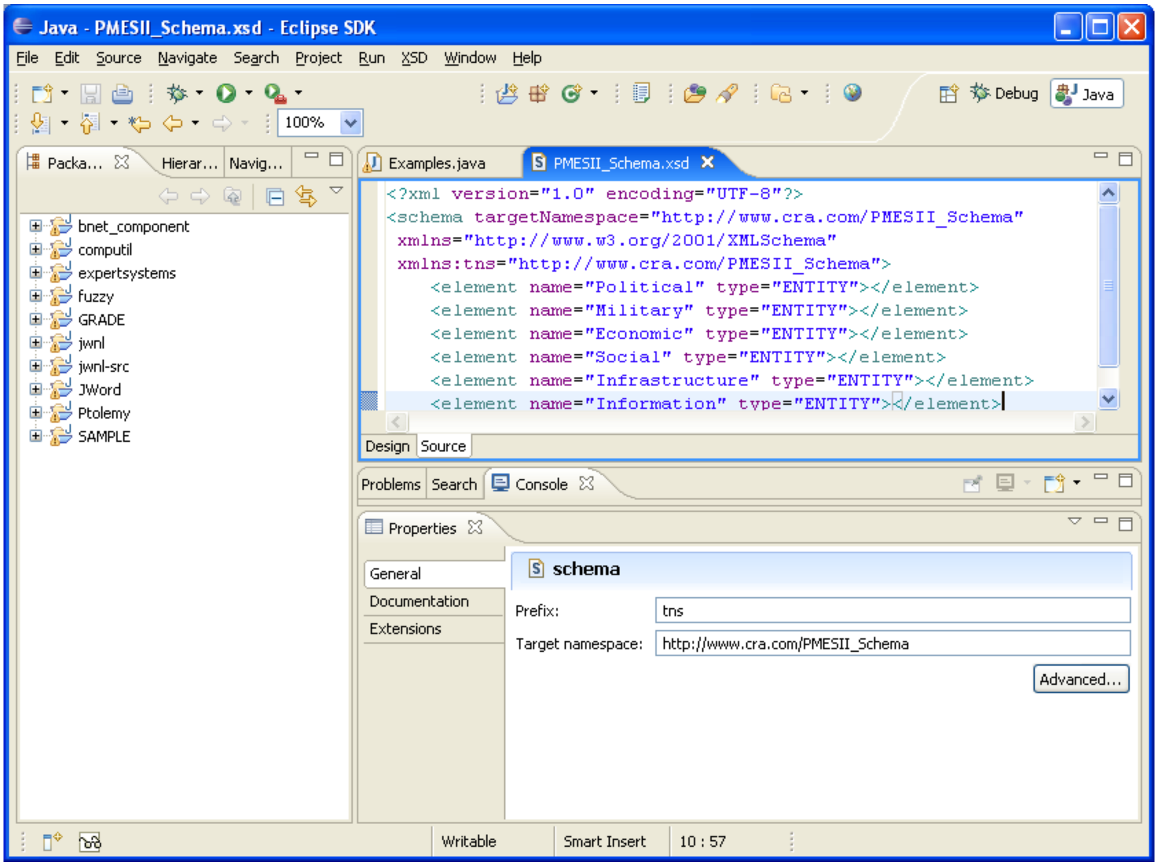
4.3.8 Ontology Evolution: Building New PMESII Models by Using and Extending Existing Models

Ontological evolution is a simple approach to building new models that addresses all of the interoperability issues described in Section 2.1. The premise is that every model built by a user / community should extend a base ontology such that it evolves over time. For instance, the core PMESII base model, schema, and ontology could declares the elements: "Political", "Military", "Economic", "Social", "Infrastructure", and "Information." A schema named "PMESII_Schema.xsl" which defines such a model and ontology is shown in the upper right frame of Figure 7. Any new models can easily extend this schema and ontology by importing it using the declared namespace. A user who wishes to investigate the Economy sub-domain can simply create a new schema by importing and extending the base PMESII model schema. The

Economy schema shown in Figure 8 does exactly that. Notice its line starting with the keyword "include" and highlighted in the figure. The "include" keyword imports all of the entities and relation definitions specified in the referenced target schema, which in this case is "PMESII_Schema.xsd". Every new PMESII model and schema that a user builds could follow such a process. The end result would be an evolving ontology that captures all models built.

It is very important to note that users would not need to know XML, nor would all models correspond to an XML schema. Users can easily be presented with a pallet in the GRADE IDE for selecting components when building a new PMESII model. The XML file and schema creation can then be done automatically in the background in response to user selections. Essentially users would be using components from previously built PMESII models when constructing new models. Any time they created a new component, this would then be added to the repository of available components.

One possible issue resulting from such a scheme is that users will create redundant components and models. A simple resolution is the use of namespaces to disambiguate between models. Each user or community could have their own namespace.



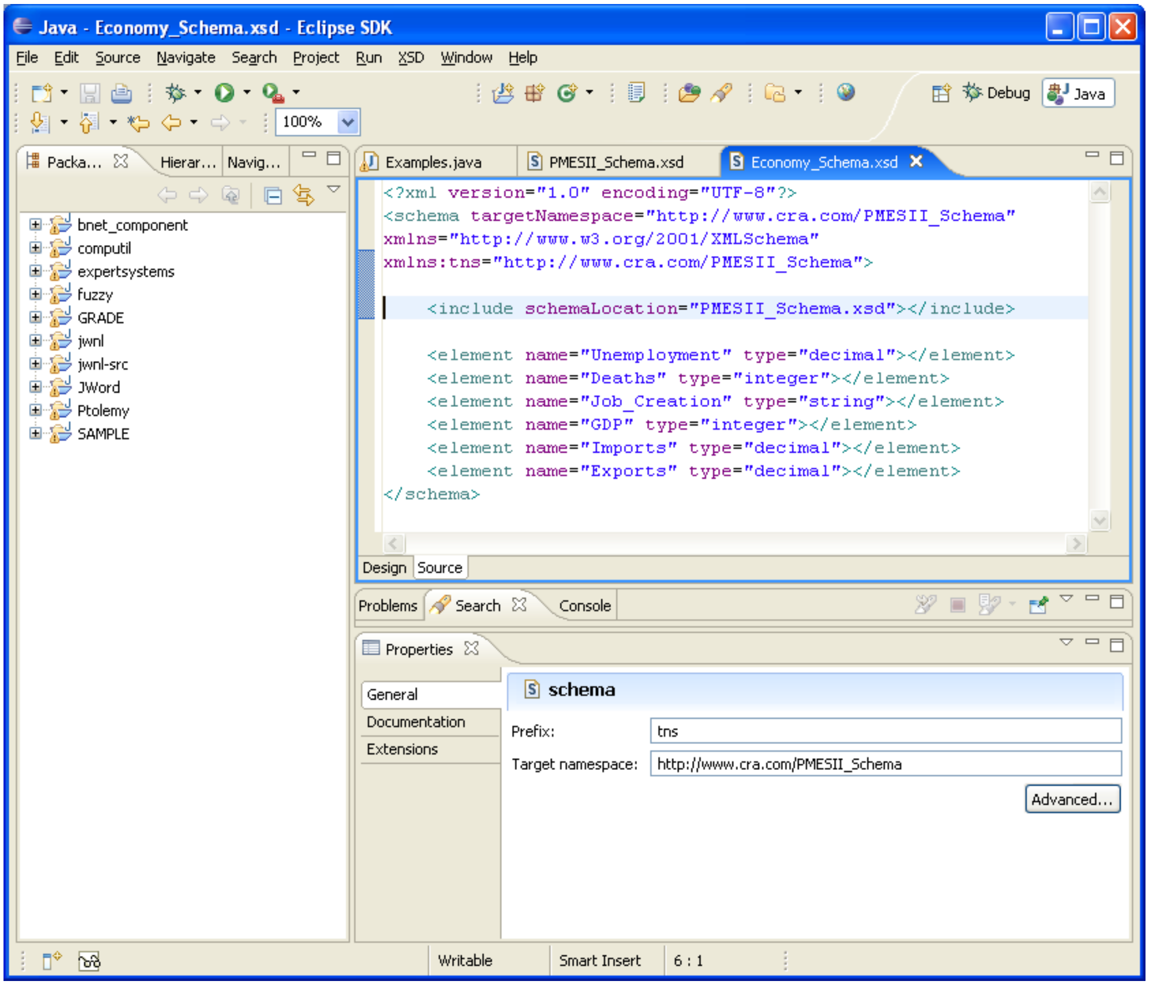
```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="http://www.cra.com/PMESII_Schema"
 xmlns="http://www.w3.org/2001/XMLSchema"
 xmlns:tns="http://www.cra.com/PMESII_Schema">
    <element name="Political" type="ENTITY"></element>
    <element name="Military" type="ENTITY"></element>
    <element name="Economic" type="ENTITY"></element>
    <element name="Social" type="ENTITY"></element>
    <element name="Infrastructure" type="ENTITY"></element>
    <element name="Information" type="ENTITY"></element>
```

schema

Prefix: tns

Target namespace: http://www.cra.com/PMESII_Schema

Figure 7: Base PMESII Schema



```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="http://www.cra.com/PMESII_Schema"
xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:tns="http://www.cra.com/PMESII_Schema">

    <include schemaLocation="PMESII_Schema.xsd"></include>

    <element name="Unemployment" type="decimal"></element>
    <element name="Deaths" type="integer"></element>
    <element name="Job_Creation" type="string"></element>
    <element name="GDP" type="integer"></element>
    <element name="Imports" type="decimal"></element>
    <element name="Exports" type="decimal"></element>
</schema>
```

Prefix: tns

Target namespace: http://www.cra.com/PMESII_Schema

Figure 8: Economy Schema Extends Base PMESII Schema

Users could provide any level of specification. Some users may drill down and even specify the type of formalisms that can be used with a model as an attribute on XML elements. A taxonomy of formalisms can also be specified so that the ontology is abstracted away from model formalisms. More specifically, users can use the same nodes for two different models, even though their edges correspond to different formalisms, e.g., rules systems vs. Bayesian networks. For instance, some entities/nodes may have an attribute "formalism" that selects what formalisms may be associated with a particular model, e.g., Bayes net, neural net, Ptolemy model.

4.3.8.1 The Benefit of an Evolved Ontology

The benefit of an evolved ontology is that it provides standardization without placing impractical constraints on the user. Models can be more easily integrated since the definitions and mappings between all possible nodes and edges in all possible models is defined (and deducible using basic logic) within the core ontological library. Communities could also publish their ontologies for inter-community collaboration. This would essentially create a Semantic Web of model ontologies (Holi & Hyvönen, 2004).

4.3.9 Federated Ontologies

A federated ontology is similar to the notion of an evolving ontology but has less structure since parts of it may be independently generated by different analysts who may or may not share components or extend a base ontology. A federated ontology could consist of any number of XML files, XML schemas, fully specified formal ontologies, and any other metadata the user chooses to provide. XML namespaces would again be used to disambiguate entities, attributes, and the relations between them. The more consistency, i.e., two models referring to the same type of entity in the same manner, within such a federation, the more powerful and accurate ontological inference can be. However, by allowing for inconsistency and redundancy, we provide the most flexibility and backward compatibility when working with different PMESII models.

4.3.10 Traditional Use of Ontologies

Ontologies have traditionally been used for knowledge representation in projects such as Cyc (Lenat & Guha, 1990) which attempts to codify all common sense knowledge. The theory is that we can perform inference using logic to find all consequents of anything specified in an ontology. For example, in the case of Cyc, we may want a computer to understand that "Judy came home" also means that "Judy is not in her car." For our purposes of model integration, we may want to infer that the edges of an Argumentation Network (Das, 2005) can map to multiple nodes of a Bayesian Belief Network (BBN) and a conditional probability table (CPT) that captures the same dynamic.

4.3.11 CPEF's Use of Ontologies

CPEF uses ontologies to perform inference on PMESII models to support their integration and interoperation. More specifically, CPEF uses ontologies to parse PMESII models, identify similarities between models, and determine how models can be connected based on the definitions of their entities, the links between entities, the overall structure of the model, and the potential values of all components. It can use XML files and schemas and the state information in PMESII models for ontological inference. Many of its methods could be termed as graph comparisons, thus ontologies are not intrinsically necessary. However, because XML files are typically used for PMESII model file persistence, and because these files can be used as ontologies, it is a natural extension to use ontological inference for this purpose.

Some of the ontology data that CPEF would leverage in integrating PMESII models is described in Sections 4.3.4 and 4.3.5. The next section describes the types of inference possible with ontologies. Section 6 provides a thorough description of where ontologies fit into the CPEF architecture along with how they are used in the processing stages of CPEF.

4.3.12 Ontological Inference

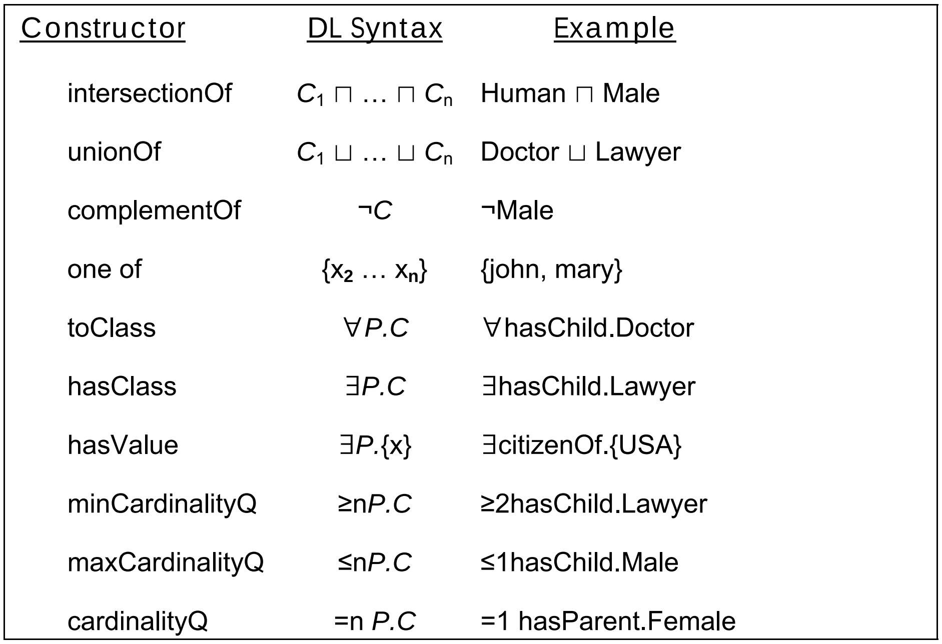
For ontological inference within CPEF we investigated the use of OWL and a number of reasoners such as Jena. OWL is an ontology specification language and is described in Section 3.4. Jena's Transitive Reasoner provides support for storing and traversing class and property lattices. This implements the transitive and symmetric properties of rdfs:subPropertyOf and rdfs:subClassOf. The reasoner is a hardwired Java implementation that stores the class and property lattices as graph structures. It is slightly higher performance, and somewhat more space efficient than the alternative of using a pure rule engines to perform transitive closure. Its main advantage is that it implements the direct/minimal version of those relations as well as the transitively closed version. The GenericRuleReasoner can optionally use an instance of the transitive reasoner for handling these two properties. This is the approach used in its default RDFS reasoner. Jena may also use Pellet, an open-source OWL-DL reasoner, as a plug-in. It is based on the tableaux algorithms developed for expressive Descriptive Logics. It accepts the SPARQL protocol and RDF query language (SPARQL) and provides a Java API to make it easy to integrate into custom systems.

Figure 29 and Figure 9 show ontology class constructors and axioms that can be used in CPEF. Both can be used to answer user queries and determine the relations between PMESII model nodes and edges. Queries may be on assertions or terminology. This implies that if we want to determine if "death" is related to "health" we may include each value in the collection of assertions, e.g., we could assert "death is a subproperty of health". In OWL it would be written:

```
<owl:ObjectProperty rdf:ID=“death”>
     <rdfs:subPropertyOf rdf:resource=“#health”/>
</owl:ObjectProperty>
```

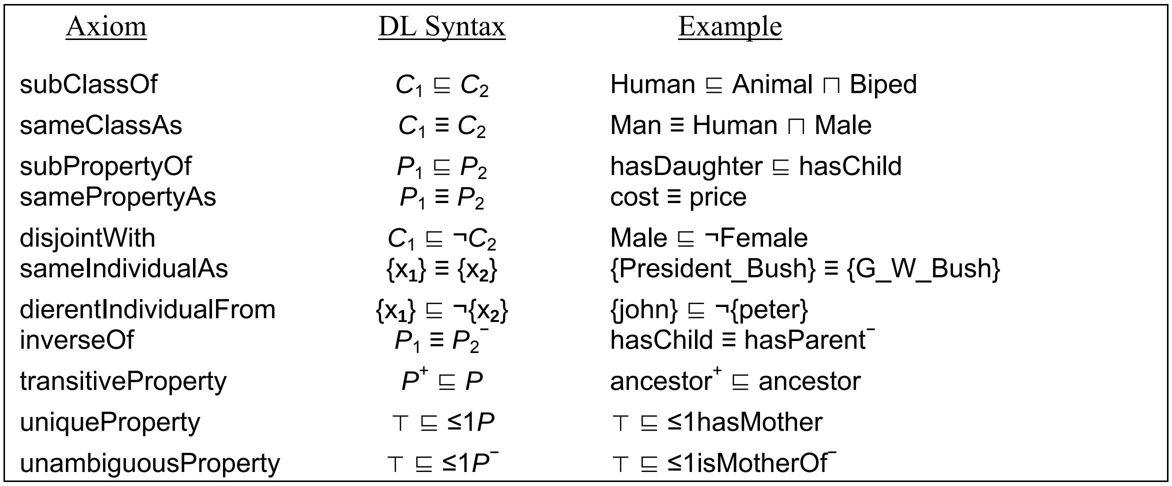
Or alternatively as part of the terminology using subclassing, "death is a subclass of health":

```
<owl:Class rdf:about=“#Death”>
     <rdfs:subClassOf rdf:resource=“#health”/>
</owl:Class>
```



Constructor	DL Syntax	Example
intersectionOf	$C_1 \sqcap \ldots \sqcap C_n$	Human $\sqcap$ Male
unionOf	$C_1 \sqcup \ldots \sqcup C_n$	Doctor $\sqcup$ Lawyer
complementOf	$\neg C$	$\neg$Male
one of	$\{x_2 \ldots x_n\}$	{john, mary}
toClass	$\forall P.C$	$\forall$hasChild.Doctor
hasClass	$\exists P.C$	$\exists$hasChild.Lawyer
hasValue	$\exists P.\{x\}$	$\exists$citizenOf.{USA}
minCardinalityQ	$\geq nP.C$	$\geq$2hasChild.Lawyer
maxCardinalityQ	$\leq nP.C$	$\leq$1hasChild.Male
cardinalityQ	$=n\ P.C$	=1 hasParent.Female

Figure 29: DAML+OIL Class Constructors



Axiom	DL Syntax	Example
subClassOf	$C_1 \sqsubseteq C_2$	Human $\sqsubseteq$ Animal $\sqcap$ Biped
sameClassAs	$C_1 \equiv C_2$	Man $\equiv$ Human $\sqcap$ Male
subPropertyOf	$P_1 \sqsubseteq P_2$	hasDaughter $\sqsubseteq$ hasChild
samePropertyAs	$P_1 \equiv P_2$	cost $\equiv$ price
disjointWith	$C_1 \sqsubseteq \neg C_2$	Male $\sqsubseteq$ $\neg$Female
sameIndividualAs	$\{x_1\} \equiv \{x_2\}$	{President_Bush} $\equiv$ {G_W_Bush}
differentIndividualFrom	$\{x_1\} \sqsubseteq \neg\{x_2\}$	{john} $\sqsubseteq$ $\neg${peter}
inverseOf	$P_1 \equiv P_2^-$	hasChild $\equiv$ hasParent$^-$
transitiveProperty	$P^+ \sqsubseteq P$	ancestor$^+$ $\sqsubseteq$ ancestor
uniqueProperty	$\top \sqsubseteq \leq 1P$	$\top \sqsubseteq \leq$1hasMother
unambiguousProperty	$\top \sqsubseteq \leq 1P^-$	$\top \sqsubseteq \leq$1isMotherOf$^-$

Figure 9: DAML+OIL Axioms

5. Recommendations for Resolving Gaps in Model Interoperability

This section presents our approaches and recommendations for maintaining, adapting, and integrating PMESII Models in the context of the interoperability gaps defined above in Section 0 and below in Table .

Table 2: Gaps and Incompatibilities

Type	**Definition**
Interface	Mismatch between the data types of different models or outputs of one model and inputs of another, e.g., Real number vs. Boolean.
Ontological	Different relationship structures, naming schemes, etc. in ontologies for different models
Formalism	Different logic and inferencing mechanisms and procedures for different models
Subdomain Gaps	Differing domains and dynamics between PMESII model dimensions, e.g., economic vs. social

5.1 Interface Incompatibility

Interface incompatibility generally refers to two or more models having different types of data for their inputs and outputs and thus not being able to interoperate without some form of data conversion. There are at least three types of interface incompatibilities:

1. I/O Format Incompatibilities: string vs. binary, real vs. integer, fixed vs. floating point, numeric vs. Boolean, incompatible scale, incompatible zero-point, date/time format, color format
2. Logical Incompatibilities: number of I/O points, e.g., 3 outputs vs. 4 inputs—RGB to CMYK is a trivial example, I/O timing, e.g., fast output vs. slow input

3. Model Persistance Format Incompatibilities: XML vs. YAML, OWL vs. RDF.

To address these issues, CPEF features an interface for encoding common translations that address each of these incompatibilities. CPEF will provide a basic set of translation functions that can learn from user interaction over time. Users can also specify these translation functions within an ontology or the XML schema of a model. Such specifications could be derived from an evolved, global ontology as described in Sections 4.1 and 4.3.8. A full-scope CPEF would feature a GUI for users to explicitly modify, add, and remove interface translation functions. A number of potential translation functions are described below in the context of what type of incompatibilities they address.

5.1.1 I/O Format Incompatibilities

It is predicted that most interface incompatibilities will fall within this category, and that most solutions will be resolved by some combination of the following:

- Normalization: mapping any value to lie between 0 and 1 relative to its minimum and maximum possible values
- Weighting: scaling a value, typically in relation to other values
- Fuzzification: randomly generating a number to lie within some constraining interval, e.g., some random number between .3 and .6
- Discretization: "binning" values according to their range and a range they must fall within – somewhat like rounding - sometimes taking their distribution into account, e.g., .5 within a range between 0 and 1 can be discretized to 1 for a range of only 0 or 1

XML schemas will typically exist for any PMESII model for the purpose of file persistence. These schemas define the elements of a model along with the possible values they can take on. XSLT can then be used along with a number of standard translation functions for integrating two models' inputs and outputs on relevant nodes and/or edges.

A base set translation functions will be included in the CPEF framework. The full-scope CPEF will feature a GUI that allows users to add, remove, and modify these functions. These functions can also adapt according to user interaction over time. The base set of functions will address most common data types, i.e., SQL data types. Some examples follow:

5.1.1.1 Number Type A to Number Type B, i.e., where A and B are one of: Real, Double, Integer, Float, Byte

Number type A will be normalized to fall within the range of B or discretized if necessary, i.e., if the range or resolution of A is greater than that of B. Context specified within the model schema or ontology will be utilized where possible.

5.1.1.2 Real to Boolean

The real will be normalized, weighted, and then discretized to true or false. Consider a neural network node "raining" that takes on values in the range 0.0 to 1.0 and has a threshold weight of -.3. It is already normalized, so we can now weight any value of the node by its threshold before mapping to a Boolean. For the node to "fire," i.e., activate other nodes, its value must be >=.8, i.e., it must exceed the weighting threshold. We would therefore map any values for the node that are <8 to false, and all others to true where firing correlates with a Boolean value of true.

5.1.1.3 Boolean to Real

Values will be fuzzified or randomized to fall within the range of the Real. For a Boolean value of true, we will generate a random number in the upper half of the range for the Real. For a Boolean value of false, we will generate a random number in the lower half of the range for the Real. Any context provided in the model schema or ontology will be used where possible, e.g., model formalism. For example, consider the task of mapping a true Boolean value to a Real number for a neural network node that takes on values between 0 and 1 and has a weighting threshold of -.3. The context of the neural network formalism can be used to deduce that the weighting threshold should be used to scale the random number generated from the Boolean value. For instance, without considering the weighting threshold, fuzzifying a Boolean value of true could result in a random Real value between .5 and 1 for the node. Taking the weighting threshold into account, we would map a Boolean value of true only to values between .8 and 1, to ensure that the node fires, i.e. firing is correlated with a Boolean value of true.

5.1.1.4 String Conversions

In cases where output are free formed Strings and no cardinality or set of possible Strings is specified in the model schema or ontology, then no support can be made. The user will have to explicitly provide some sort of translation in the ontology, schema, or through the CPEF GUI. However, in many cases, we can at least rely on the model schema to specify the number of possible Strings that can be specified for a particular value. We can then use this information to provide some level of automatic translation.

For instance, a user could create a model A with an "economy" element that takes on a decimal value between 0 and 1. Another user may create a model B that also has an economy element but allows it to take on 3 different string values: "good", "neutral", "bad". This is not only a difference in granularity of data values, but also a difference in the basic data type being used to represent the values for each economy element. We can use the XML schema for both models and an XSLT translator to map from one type of value to the other. In our example, we can normalize the integer values and then discretize them to fall within three ranges, one range being associated with each possible string value. Specifically, values for model A's economy element fall between 0 and 1, thus we can create the following mapping:

- Economy < .33 = bad
- Economy >.33 <.66 = neutral
- Economy >.66 = good

5.1.2 Logical Incompatibilities

In some cases, model A may have more outputs than model B has inputs or vice versa. When integrating models, we therefore need methods for addressing these situations. For an overabundance of values we can simply use some form of aggregation. Again, a model schema or ontology can specify how this aggregation should be performed, or the user could specify this through the CPEF GUI. In the case where we have only one value but must map to more than one, we can simply duplicate the value or partition it according to any context provided in the model schema or ontology. Duplication will be the default unless the user specifies otherwise.

In some cases, the rate of output and/or input can differ. It is believed that this will not cause a problem in most cases, however the user may want to specify behavior in the model schema or ontology in order to affect more precise or specific behavior in the integrated model.

5.1.3 Model Persistence Format Incompatibilities

In essence, this issue really mirrors the greater task of integrating PMESII models. The existence of a standard schema or ontology for PMESII models would immediately resolve this issue. Unfortunately we can not depend on such a standard or adherence to it. A partial solution may be to evolve or derive a standard schema or ontology as described in sections 4.1 and 4.3.8. In either case, most effective solutions will entail the use of XML and XSLT for the translation of one model format to another.

5.2 Ontological Incompatibility

Ontological incompatibility refers to two models having different structures including the entities they specify and the relationships between them. For instance, a rules system model may have several pairs of nodes connected by one edge (precedent and consequent) where as a Bayesian net will typically have more of a tree structure. Nodes can have different names, graphs can be directed or undirected, and two models representing the same system can be at different resolutions and thus include a different number of nodes and edges. The principle issue of this incompatibility is determining which entities, nodes, or edges in different models should map to one another for interoperation. CPEF is designed to leverage several heuristics and methods for inferring these mappings as described below.

5.2.1 Syntactic Heuristics

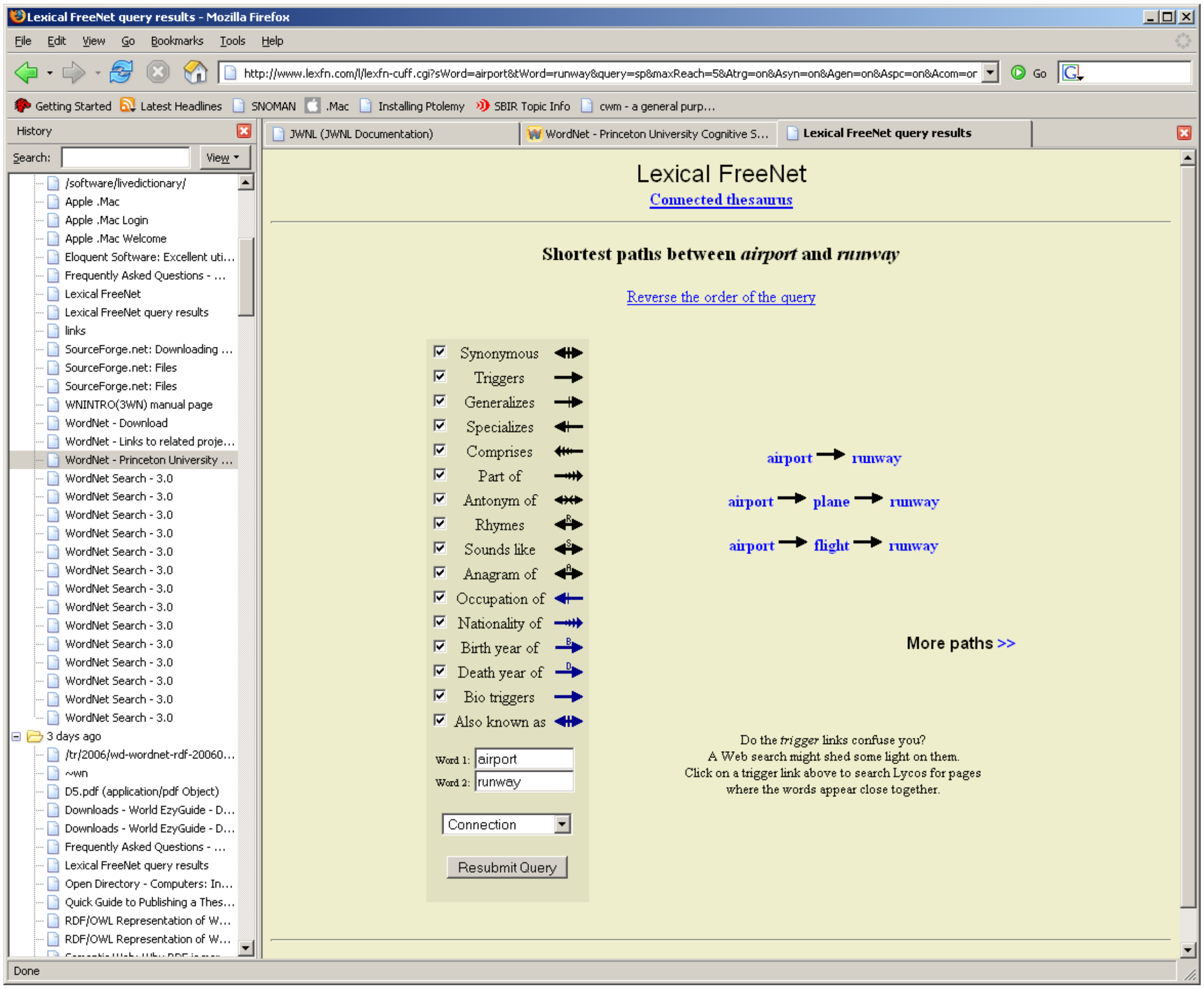
The labels and descriptions of nodes and edges in differing models can be compared based on their raw string content. If these String components match then the nodes or edges may be a match as well. For instance, "runway16" may map to "runway". A threshold for how many characters must match to infer a string match must be specified. CPEF will contain a default that can be modified by the user and adapted over time (or learned) based on user interaction. This type of matching can also include matching nodes/edges based on the range, cardinality, and other attributes of their possible values.

5.2.2 Semantic Heuristics

Nodes and edges from different models can be compared based on the semantics of their labels, descriptions, and any other textual metadata specified in a XML file, XML schema, and/or ontology. Elements from different models that have a semantic similarity can then be mapped to one another for model integration. For instance, a node with the name "airport" in one model may be mapped to a node with the name "runway" in another model based on the semantic similarity of their labels. Semantic similarity is determined by the relations between two words as derived from statistical usage, ontologies, thesaurus, dictionaries, etc. There are both Service Oriented Architectures (SOA) and API specifications for this purpose including WordNet (Al-Halimi et al., 1998) and Lexical Freenet.

WordNet was begun in 1986 and is based on Prolog, but RDF translations of it and a Java interface are now in the public domain. The database now contains about 150,000 words organized in over 115,000 synonym sets for a total of 207,000 word-sense pairs. The public domain toolkit JWNL (Java WordNet Library) provides an API to access this capability.

Lexical Freenet is a natural language processing (NPL) implementation that uses WordNet for semantic comparison. Figure 10 shows the web interface for Lexical Freenet matching "runway" to "airport" based on semantic similarity. How to use these and other enabling technologies is described further in Section 6 in the context of a system design specification.


Lexical FreeNet query results - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://www.lexfn.com/l/lexfn-cuff.cgi?sWord=airport&tWord=runway&query=sp&maxReach=5&Atrg=on&Asyn=on&Agen=on&Aspc=on&Acom=or
Go
Getting Started
Latest Headlines
SNOMAN
.Mac
Installing Ptolemy
SBIR Topic Info
cwm - a general purp...
History
Search:
View
/software/livedictionary/
Apple .Mac
Apple .Mac Login
Apple .Mac Welcome
Eloquent Software: Excellent uti...
Frequently Asked Questions - ...
Lexical FreeNet
Lexical FreeNet query results
links
SourceForge.net: Downloading ...
SourceForge.net: Files
SourceForge.net: Files
WNINTRO(3WN) manual page
WordNet - Download
WordNet - Links to related proje...
WordNet - Princeton University ...
WordNet Search - 3.0
WordNet Search - 3.0
WordNet Search - 3.0
WordNet Search - 3.0
WordNet Search - 3.0
WordNet Search - 3.0
WordNet Search - 3.0
WordNet Search - 3.0
WordNet Search - 3.0
WordNet Search - 3.0
WordNet Search - 3.0
WordNet Search - 3.0
WordNet Search - 3.0
WordNet Search - 3.0
3 days ago
/tr/2006/wd-wordnet-rdf-20060...
~wn
D5.pdf (application/pdf Object)
Downloads - World EzyGuide - D...
Downloads - World EzyGuide - D...
Frequently Asked Questions - ...
Lexical FreeNet query results
Open Directory - Computers: In...
Quick Guide to Publishing a Thes...
RDF/OWL Representation of W...
RDF/OWL Representation of W...
JWNL (JWNL Documentation)
WordNet - Princeton University Cognitive S...
Lexical FreeNet query results
Done

Lexical FreeNet

Connected thesaurus

Shortest paths between *airport* and *runway*

Reverse the order of the query

☑ Synonymous
☑ Triggers
☑ Generalizes
☑ Specializes
☑ Comprises
☑ Part of
☑ Antonym of
☑ Rhymes
☑ Sounds like
☑ Anagram of
☑ Occupation of
☑ Nationality of
☑ Birth year of
☑ Death year of
☑ Bio triggers
☑ Also known as

Word 1: airport

Word 2: runway

Connection

Resubmit Query

airport → runway

airport → plane → runway

airport → flight → runway

More paths >>

Do the *trigger* links confuse you?
A Web search might shed some light on them.
Click on a trigger link above to search Lycos for pages
where the words appear close together.

Figure 10: Semantic Matching of Airport and Runway

5.2.3 Relation Mapping

Relation mapping can be used to address ontological incompatibility by mapping nodes from one model to nodes of another based on their relations (how they are connected) within their individual models. With this information, we can then suggest potential mappings between nodes of different models based on the similarity of their relations within their respective models. Consider the nodes α of model A and β of model B. Though these nodes may have very different names, they may have very similar relations. For example, both could influence five other nodes and be influenced by four other nodes. Based on their similarity, we may be able to deduce that these nodes can be mapped together for model integration. It is important to note that relations encompassing a node are not merely all of its incoming and outgoing edges; they also include features identifying how the node affects any other nodes in the model. For BBNs, that would be analogous to performing a sensitivity analysis on a node. It is possible that two nodes from different BBNs may be capable of being mapped to each other based on each having a similar degree of influence within their respective models. While this approach should rarely be used to automatically draw links, it can be used to make effective recommendations.

5.2.4 Model Node Aggregation

Model aggregation can be used to address ontological incompatibility by identifying how sets of nodes in different models with differing cardinalities may be mapped to one another. It may be the case that a node α in model A maps to a subset of nodes N in model B, resulting in incompatible ontologies as in the two frames of Figure 11. For example, consider α to be the node airport and N to be the subset of nodes runway, plane, radar, and air traffic control. The question is, which nodes should airport be mapped to for model integration? We can use the semantic similarity of the labels on the nodes of N (e.g., interfacing with WordNet for ontological inference) to aggregate N into one meta-node: airport' as in the bottom frame of Figure 11. Using semantic similarity between airport and the constituent nodes of airport', we can then infer that these two entities should be mapped for model integration. More specifically, the inputs to airport could potentially be mapped as inputs to all nodes of airport'.

We can also infer the pairing of airport and airport' using relation mapping. To continue the example, consider relations between airport and the other nodes of model A, and airport' and any remaining nodes of model B. If their relations are similar, then airport could be a candidate for integrating with all of the nodes of airport'. For instance, both airport and airport' could be connected to the nodes passenger, ticket, and pilot. With ontological inference, we can see that the relations of airport and airport' are similar within their respective models (even though the relations of airport are not similar to the relations of any of the constituent nodes of airport'). We can then deduce that these nodes should be mapped for model integration.

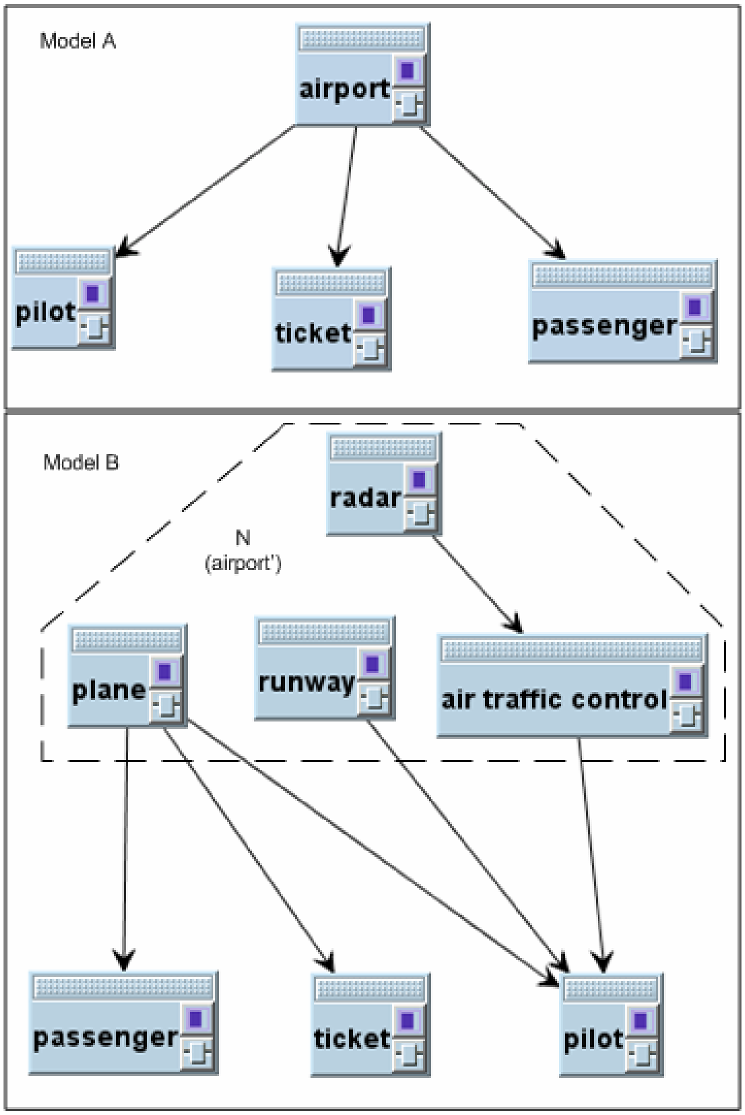


Figure 11: Model Node Aggregation Example

5.3 Formalism Incompatibility

5.3.1 Established Formalism Mappings

There has been a great deal of research on mapping between different algorithm formalisms and there are a number of established standards. Table 3 shows a matrix where the following formalisms appear in the outer cells of both the X and Y axis: Bayesian Probability, Dempster-Shafer, Fuzzy Logic, Possibilistic Theory, Certainty Factor, and Symbolic Dictionary. Each internal cell denotes the mechanism used for mapping between associated formalisms on the outer cells of the X and Y axis. Note that the mechanism for mapping from X to Y may not necessarily be the same mechanism used for mapping from Y to X. Highlighted cells represent established mechanisms for mapping between different formalisms, while non-highlighted cells represent our recommendations for mapping approaches. In other words, there are known and established algorithms for mapping between the formalisms that are joined by a highlighted cell.

Table 3: Formalism Mappings

	Bayesian Probability	Dempster-Shafer	Fuzzy Logic	Possibilistic Theory	Certainty Factor	Symbolic Dictionary
Bayesian Probability	X	Generalization	Membership Degree Interpretation of Probabilities	Transformations based on Consistency Principles	Mapping from Probabilities to Certainty Factors	Probability-to-Symbolic Mapping
Dempster-Shafer	Bayesian Approximation (Transferable Belief Model)	X	Via Bayesian Approximation	Via Bayesian Approximation	Via Bayesian Approximation	Belief Value-to-Symbolic Mapping
Fuzzy Logic	Normalization of Membership Degrees	Via Normalization	X	Possibility Measure Interpretation of Membership Degrees	Mapping from Membership Degree to Certainty Factors	Membership Degree-to-Symbolic Mapping
Possibilistic Theory	Transformations based on Consistency Principles	Belief Interpretation of Possibility Measures	Membership Degree Interpretation of Possibility Measures	X	Mapping from Possibility Measures to Certainty Factors	Possibility Measure-to-Symbolic Mapping
Certainty Factor	Normalization	Via Normalization	Via Normalization	Via Normalization	X	Certainty Factor-to-Symbolic Mapping
Symbolic Dictionary	Symbolic-to-Probability Mapping	Symbolic-to-Belief Value Mapping	Symbolic-to-Membership Degree Mapping	Symbolic-to-Possibility Measure Mapping	Symbolic-to-Certainty Factor Mapping	X

5.3.2 Using XML Schemas and Ontologies for Formalism Mapping

Ontologies can explicitly identify mappings between formalisms in the attributes of edges within a PMESII network model. For formalisms such as Bayesian networks and argumentation networks, relations correlate with the edges between nodes. We can add an attribute to each edge in a Bayesian network XML schema declaring a “causes” relationship between a parent and child node it connects. Further, we can specify that types of edges declaring a “causes” relationship should be mapped to the “entails” relationship of a rule-based model. When mapping from a Bayesian network model to a rule based model, we can then infer from the ontology that a “cloudy” node in a Bayesian network that “causes” a “rain” node should be mapped to a rule that has the variable “cloudy” as a precedent and “rain” as a consequent.

5.4 Sub-Domain Gaps

Sub-domain Gaps can be addressed by learning ontologies and ontological evolution where the relationships between models are implicitly specified as models are built. These approaches are described in section 4.1 and 4.3.8. Mixed initiative approaches can also be used throughout the CPEF structure to address all of the interoperability gaps including differences in sub-domains. For instance, suggestions could be made as to what nodes and/or edges should be related between two models. The user could then accept, edit, or ignore these suggestions. One simple mixed initiative approach would be reinforcement learning (Kaelbling, Littman, & Moore, 1996) guided by these user selections. If a user accepts a suggestion, the system could increase the number of related suggestions. If a user rejects a suggestion then the system should learn not to make similar suggestions in the future. Even devoid of other heuristics, CPEF can store historical information as to what input and output types the user typically maps together and offer these mappings as suggestions for subsequent model integration.

6. Recommendations for System Design and Enabling Technologies

This section provides a design specification for a system that implements the approaches outlined in the previous section. Included here is a description of the current CPEF architecture along with recommendations for a full-scope implementation, enabling technologies, and the procedural steps during model integration.

6.1 CPEF System Architecture

Figure 12 shows the CPEF System Architecture in the context of integrating with a Service Oriented Architecture (SOA) for referencing PMESII model repositories, lexical databases, ontology databases, and related data resources. In this section we present recommendations for the design of the CPEF system. Each subsection provides an overview of a CPEF architecture component along with enabling technologies that can be used for its implementation. We interleave the existing CPEF framework with recommendations for a full-scope implementation.

CPEF can be implemented within a larger Service Oriented Architecture concept. Specifically, its use of WordNet and related lexical databases for semantic analysis and existing models and ontologies for ontological inference can be programmed as a web services application. The PMESII Model Repository in the upper left of Figure 12 represents data storage of PMESII model XML files, XML schemas, and ontologies as described in Section 4.3. This information will be used primarily by the Inference Engine as described below. The Integrated Systems shown in the upper right of the system architecture represents lexical databases, ontologies, and also intelligence data that may be used by the Model Analyzer.

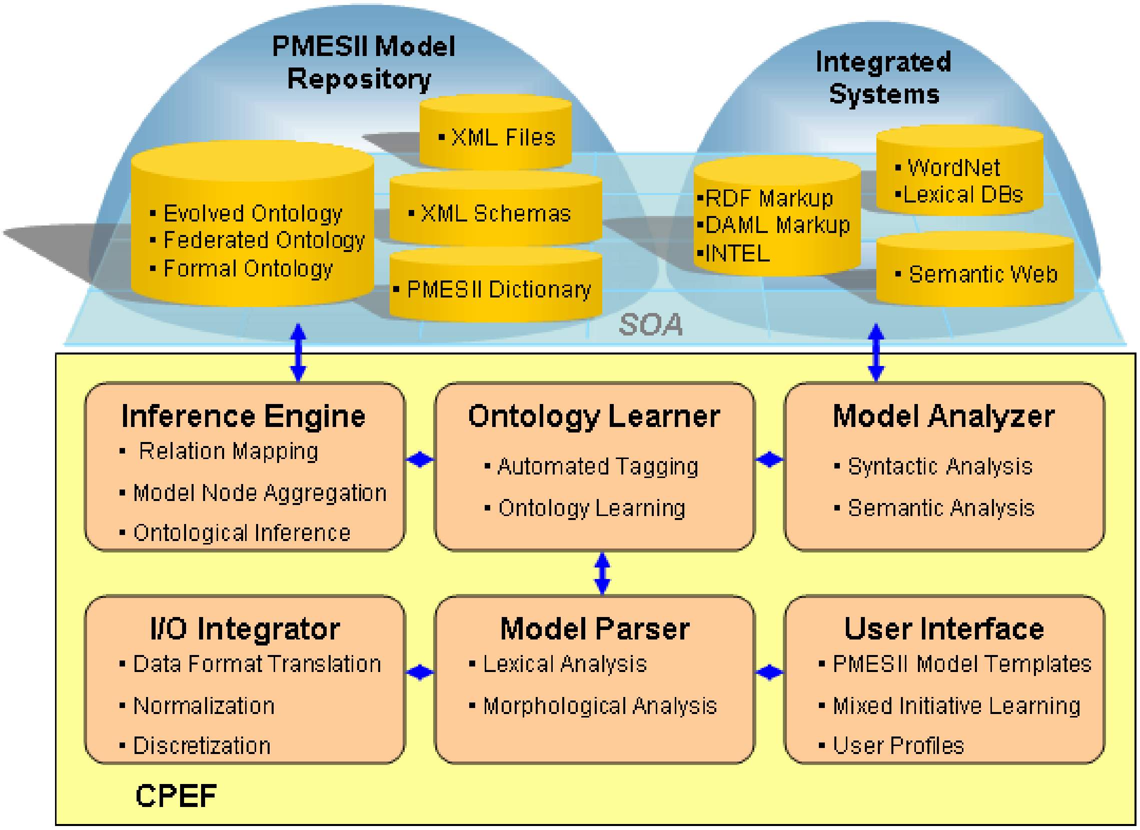
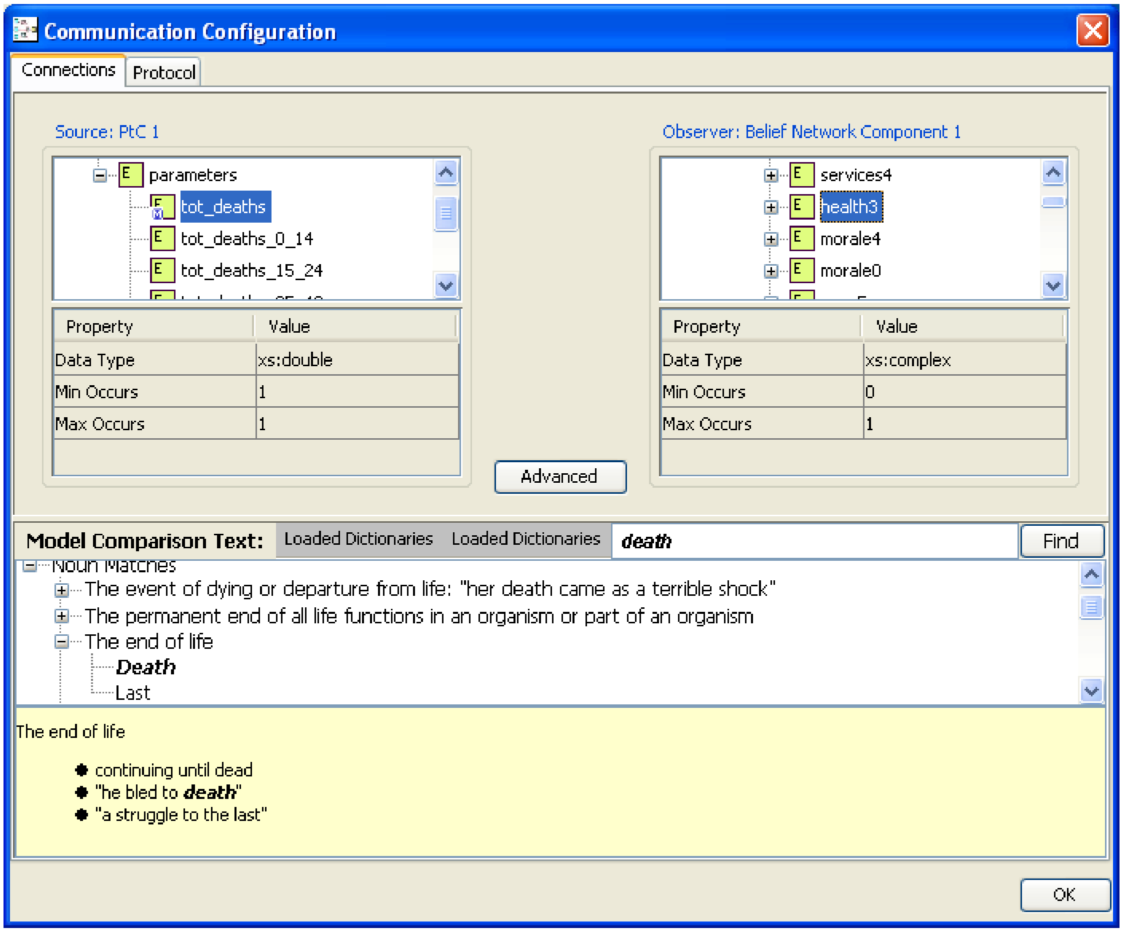


Figure 12: CPEF System Architecture

6.1.1 User Interface

The user interface for CPEF is a sub-component of the PMESII IDE provided by GRADE. Within the prototype it consists of one window where two models are displayed side by side and suggestions are made for how their nodes, attributes, inputs, and outputs can be mapped for model integration. The user interface for the CPEF prototype is shown in Figure 13. The bottom frames of the CPEF prototype show the results of a semantic analysis in order to explain the suggestions being made for model mapping and integration. In Figure 13 the bottom frame is showing the results of a semantic analysis of "death" which corresponds to the nodes in the health model (shown in the upper left frame of the figure) for the number of deaths over discrete periods of time.



Source: PtC 1

- parameters
  - tot_deaths
  - tot_deaths_0_14
  - tot_deaths_15_24

Property	Value
Data Type	xs:double
Min Occurs	1
Max Occurs	1

Observer: Belief Network Component 1

- services4
- health3
- morale4
- morale0

Property	Value
Data Type	xs:complex
Min Occurs	0
Max Occurs	1

Model Comparison Text: *death*

Noun Matches
- The event of dying or departure from life: "her death came as a terrible shock"
- The permanent end of all life functions in an organism or part of an organism
- The end of life
  - ***Death***
  - Last

The end of life

- continuing until dead
- "he bled to ***death***"
- "a struggle to the last"

Figure 13: CPEF Prototype

6.1.1.1 PMESII Model Templates

In a full-scope CPEF implementation, there will be a pallet of nodes, links, and default models that the user can choose from in creating new PMESII models. Not only will this expedite the process of model construction, but it also facilitates the use and evolution of a PMESII ontology as described in Section 4.3.8. Such a mechanism will resolve many of the issues of model integration, adaptation, and interoperability.

6.1.1.2 Mixed Initiative Learning

Section 4.1 describes the concept of mixed initiative learning. The driver of this mechanism will be user interactions with the CPEF interface. For instance, CPEF may suggest the user map the "death" nodes of the model on the left of Figure 13 with the health nodes of the model on the right. This would be done by highlighting both nodes and is shown in the figure. However, the user may actually decide to use the number of deaths in one model as an input to the morale of the people in the other model. To do so the user would unselect the "health" node and select the "morale" node of the model on the right. In doing so, the CPEF system can store this interaction and weaken the strength of its recommendations for mapping "death" to "health" within any models. If the user subsequently maps "death" to "morale" in other models, the CPEF system can learn this new mapping and start providing it as a recommendation.

6.1.1.3 User Profiles

Section 4.2 describes the concept of collaborative filtering. This is the technology used to make movie recommendations on NetFlix and related websites and services. The CPEF system can store user profiles so as to accurately aggregate user statistics, learn from use cases, and improve recommendations for model integration over time. In addition, user profiles can store settings specific to a user such as the types of nodes and links typically used to expedite model construction and integration.

6.1.1.4 Enabling Technologies

Our in-house GRADE component can provide all JAVA APIs necessary for the CPEF user interface. There are also a number of graphing and modeling tools and APIs that can be used.

6.1.2 Model Parser

The model parser is responsible for loading PMESII model files and parsing them according to their native format. It should have a battery of file I/O primitives based on common file types such as XML, XSD, and RDF. It will also have a simple heuristic that checks files for specification type, e.g. XML, RDF, OWL, in order to determine what heuristics and algorithms should be used for later processing. The final result of model parsing will be a parse tree that CPEF can refer to and manipulate during other processing stages. The Model parser will consist of a number of sub-processing components to support low level natural language processing (NPL) and the operation of other CPEF components. These will process text on node and edge labels and descriptions and can include grammar specific repair heuristics, full minimal distance parsers, skipping and maximal coverage parsers, and connectionist parsers. A basic set of parsing components and stages are summarized in the subsections below.

6.1.2.1 Morphological Analysis

Morphological analysis has to do with mapping from one form of a word to another along dimensions such as part of speech or plurality for purposes of semantic analysis. For instance, a lexical dictionary may contain an entry for "run" but not "running". In order to perform semantic analysis with that dictionary, a system must first derive the word "run" from "running". Morphological analysis does this by indexing a record of statistical information on how words are used in order to map from one to another. A more sophisticated example may be mapping "method" to "methodology", or "sub-domain" to "subdomain".

6.1.2.2 Lexical Analysis

The Model Parser should contain one or more sub-parsing routines for lexical analysis of model component labels, descriptions, and any other specified metadata. Lexical analysis strips words of punctuation, numbers, and other symbols that would interfere with other heuristics and algorithms used in CPEF. This is a standard component of NPL architectures. Several components of CPEF including the syntactic and semantic analyzers rely on parse trees containing non-compound, contiguous strings of characters for analysis. This is standard among NPL architectures. For instance, a user may employ terminology that is specific to a particular mission, or they may use numbers in certain node labels. To use our common example of runway and airport, the user may have a model with a node labeled "runway13" since there are 13 runways and he is specifically referring to one of them. When integrating that model with another that has a node labeled "airport", in order to syntactically and semantically compare those two labels, we need to truncate "runway13" to "runway". This is a very simple process, however can become more complicated in particular instances of punctuation such as "it's".

6.1.2.3 Enabling Technologies

There are several open source and commercial toolkits for natural language processing. One popular one is aptly named Natural Language Toolkit and is available on Source Forge at this web location: http://nltk.sourceforge.net/. It features several parsers that can be used within the model parser component, and even features GUIs for examining their output and performance.

6.1.2.3.1 WordNet

WordNet is an open source project out of Princeton which consists of a lexical database that can be used for semantic analysis. **JWNL** is the java API for WordNet and includes a number of primitives for morphological analysis. The CPEF prototype uses JWNL and a local WordNet dictionary, however WordNet can also be used as a web service within a Service Oriented Architecture.

6.1.2.3.2 JWord

JWord is an open source Java API that integrates with WordNet using JWNL. The benefits of JWord are its open plug-in architecture. Additional dictionaries, including a PMESII dictionary and ontology, can easily be integrated by writing a "PMESII-Driver". The CPEF prototype integrates JWord with GRADE and uses not only WordNet but also the Roget thesaurus.

6.1.3 Inference Engine

The Inference Engine is largely responsible for performing the analysis described in Section 2.1.2, however, it can be used to address all model incompatibilities and functionality gaps. It can use XML files, schemas, and formal ontologies as follows:

- **Interface Incompatibility:** validate data types and interfaces using XML schemas or infer validation from node relations in order to provide mappings between data types and data translation for input/output pairing
- **Ontological Incompatibility:** perform reasoning based on the structure and specification of models for integration as described in Section 2.1.2
- **Formalism Incompatibility:** refer to a taxonomy of formalisms and instructions for how they are mapped within an ontology, execute the mappings specified in an XML schema, or infer formalism mappings by comparing similar relations in the structures of PMESII models
- **Subdomain Gaps:** refer to XML and ontology specifications and model relations such as common parents or children to see how different domains can be mapped to one another.

The inference engine can also be used to support querying and reasoning with single models and networks of heterogeneous models. Most first and second order logics can be executed for answering such questions as:

- "What entities influence the morale of the people?"
- "What number of civilian deaths is required to influence the people to riot within a particular region?"
- "Is there an entity that influences both economy and military infrastructures?" i.e., eliminating such an entity can achieve the goals of targeting both economy and military infrastructures.

6.1.3.1 Enabling Technologies

6.1.3.1.1 Jena

Jena is a Java framework for building Semantic Web applications. It provides a programmatic environment for RDF, RDFS and OWL, SPARQL and includes a rule-based inference engine. Jena is open source and grown out of work with the HP Labs Semantic Web Program. The Jena Framework includes:

- An RDF API
- Reading and writing RDF in RDF/XML, N3 and N-Triples
- An OWL API
- In-memory and persistent storage
- SPARQL query engine

6.1.3.1.2 Pellet

Pellet is an open-source reasoner that works with OWL-DL. It accepts the SPARQL protocol and RDF query language (SPARQL) and provides a Java API to make it easy to integrate into custom systems. According to the literature, Pellet is a complete and capable reasoner with very good performance, extensive middleware, and a number of unique features.

6.1.4 Model Analyzer

The model analyzer is responsible for syntactic and semantic analysis of any metadata provided in PMESII models as described in Section 5.2. This includes text labels and descriptions on nodes, edges, and attributes. It is primarily used for resolving ontological incompatibilities, although it can also be used in other processing stages. It is important to note the distinction between model analysis and ontological inference. Within this document, related presentations, and the CPEF system, ontological inference refers to logical operations performed on the structures of PMESII models. Model analysis may use lexical dictionaries which are themselves ontologies, however it primarily operates on the text internals of PMESII models rather than their structure or data specific entries such as data type and cardinality of values.

The model analyzer will be used primarily for mapping nodes and edges from one model to another for integration based on one or more of the following:

- Part of speech (POS)
- Plurality
- Semantic similarity of text labels and/or descriptions
- Common usage patterns
- Semantic relationships (antonym, hypernym, hyponym, etc.)

The model analyzer will also be used in resolving formalism incompatibilities by investigating semantic relations between words, e.g., “entails” in sentence structure to be related to “entails” in a rules system, and addressing subdomain gaps by investigating potential hyponyms and hypernyms, e.g., how two domains may have a common parent, even if necessary to recurse back to the highest levels of a PMESII hierarchy.

6.1.5 Ontology Learner

The ontology learner observes user interactions, consumes parsed models, and clusters the information available in model repositories to derive ontologies and perform automated tagging and annotation. The resulting ontologies are also stored within the model repository. The premise for doing this is that much of the analysis necessary for model integration can be done during model construction and stored for later processing. In other words, when a user creates a model, some level of analysis can be performed on it, even if the user is not currently engaged in model integration. The output of that analysis can be a learned ontology, which is used later when the user _is_ engaged in model integration.

The use of ontologies in CPEF is described in Section 4.3 along with how ontology learning and automated tagging can take place. The more metadata specification the better, however CPEF is designed to handle variable levels of specification including none at all. However, with ontology learning, ever model construction adds to the ontology base of CPEF and can be used for model integration.

6.2 CPEF Processing Stages

This section presents a high level overview of the processing stages of CPEF in a linear sequence. These stages assume the implementation of the architecture shown in Figure 12.

- Users construct models using templates and/or a pallet of components available from existing models and manual plotting nodes, edges, and their relations. This is done via the **User Interface** and related mechanisms.
- Users can subsequently load multiple models for integration and use that may differ in structure, domain, and formalism, i.e., may have incompatibilities described in Section 2.1.
- The **Model Parser** of CPEF parses the models into a tree structure and prepares it for syntactic and semantic analysis. An initial schema, metadata, and ontology lookup is performed as some users may explicitly specify how models should be integrated.
- The **Model Analyzer** takes the parse tree of both models and compares them based on semantic similarity to generate an initial list of potential node, edge, input, and output pairings. It performs syntactic and semantic analysis using lexical databases shown as **Integrated Systems** in Figure 12.

- The **Inference Engine** takes the output of the **Model Analyzer** and refines the list of potential node, edge, input, and output pairings using ontological inference as described in Section 4.3 (and more specifically Section 4.3.12) and 5.2. It uses a database of existing models and ontologies in the **PMESII Model Repository** as a point of reference for its processing.
- The recommendations for model integration are presented to the user in the CPEF **User Interface**. The user accepts, rejects, or modifies the recommendations for model integration provided by CPEF. These interactions are fed into algorithms for mixed initiative learning and the **Ontology Learner**.
- The **I/O Integrator** takes the selections made by the user for model integration and attempts to create a mapping or translation between the data types of inputs and outputs of the integrated models. This processing uses the approaches for resolving interface incompatibilities as described in Section 5.1. If there is any conflict, the user is presented with selections for resolving the interface incompatibilities.
- The final results are parsed by the **Ontology Learner** and any necessary modifications are made to the **PMESII Model Repository** for subsequent processing.

7. Reasoning with Heterogeneous but Integrated PMESII Models

At a fundamental level, reasoning with PMESII models is performed to determine what inputs, to what entities, generate what outputs. Section 2.2 describes some of the issues involved in reasoning with PMESII models while Section 4.3.12 describes how reasoning can be performed with ontological inference. Here we describe some of the implications of reasoning with a network of potentially heterogeneous PMESII models, i.e., more than one interconnected models that have different formalisms e.g. Ptolemy model, Bayesian network, neural network, to identify leverage points that represent opportunities to influence capabilities, perceptions, decision making, and behavior. This is essentially the reason for model integration in the first place.

7.1 Formalism Issues

A difference in formalisms of two interconnected models affects not only model integration, but also how inference and reasoning can be performed. Specifically, the question of how activation is propagated through a network is contingent on the formalism of a model. This may not be an issue for certain interconnected models, e.g., for neural networks, outputs are always "on" or "off", for rules systems outputs are "true" and "false", both of which can be mapped fairly easily to one another. However, for some models, such as Bayesian networks, the propagation of evidence and probability requires aggregation algorithms that do not translate to other formalisms.

Many of the typical inference techniques for determining how inputs to one node of a model will influence the outputs of another node involve some form of backward chaining. Some formalisms preclude such techniques because of the manner in which their inputs and outputs are processed. To use backward chaining to determine what inputs to one model produce what outputs of another model, both models must have the same formalisms, and that formalism must allow backward chaining. CPEF therefore includes designs for reasoning models that do not entail backward chaining as described below.

7.2 Approaches for Reasoning with Integrated Models

Here we provide two methods for reasoning with integrated and heterogeneous models that ***do not*** share a common formalism. These methods can be used for reasoning with individual PMESII models as well, however their strength is in their robustness and general applicability when working with networks of models.

7.2.1 Monte Carlo Simulation

Monte Carlo simulation with PMESII models involves randomly generating inputs, to randomly selected nodes of networked models and observing the outputs. The goal is to achieve a uniform sampling of the space of all possible inputs in order to derive a general description of system dynamics. One of the advantages of this approach is that it can be performed rapidly, however the more simulations, the more accurate the description of system dynamics. This has to do with the law of large numbers and probability.

7.2.2 Sparse-Grid Simulation and Data Mining

Sparse-grid simulation iterates through a subset of all possible networked model inputs and observes each networked model output. The subset is a discretization of all possible networked model input values. For instance, if one node can take on any decimal value between 0 and 1 then it might be discretized to the set of values [.0, .2, .4, .6, .8, 1.0].

The outputs of all simulations can be stored in a database and then data mining can be performed. In general, the input values for a simulation will be the key to its entry in the database, and the outputs will be the values within its entry. Data mining algorithms such as association rules can be computed in linear time to determine correlations between inputs and outputs between models. The result of such algorithms will be a description of system dynamics, what entities represent the greatest influence on the system, and how they influence that system.

This approach can be parallelized and performed using grid computing architectures, however in most cases it will incur a high cost in computation time. The advantage of this approach is its comprehensive investigation of system dynamics. The finer grained the discretization of possible networked model inputs, the more accurate and detailed the description of system dynamics.

8. Appendix: Uncertainty, Classification, Approaches, and Gaps Resolution

Ignorance can be generally characterized as lack of information. Different types of knowledge may be lacking in a knowledge base of PMESII models and consequently give rise to different kinds of uncertainty. Here we present a survey of certain common forms of ignorance and of the mathematical techniques that have been suggested to quantify ignorance. We then discuss interrelationships among these techniques.

Ignorance can broadly be subdivided into the following three categories (Bonissone and Tong, 1985; Smets, 1991):

- **Incompleteness**: Incomplete information refers to cases where the value of a variable is missing. Consider a database of battlefield reports that should include information about detected unit position, category (friendly, neutral, and hostile), and type (tank, armored personnel carrier, Humvee …) for each report. If, for example, the value of the variable representing the type of a detected hostile unit at a certain location is missing from a report, then the information in the report becomes incomplete. But the rest of the information in the report is precise and certain, i.e., a hostile unit has been detected at a location.
- **Imprecision**: Imprecise information refers to cases where the value of a variable is given, but not with enough precision. Suppose the value of the variable representing unit type is tank or "Humvee" of a detected hostile unit at a certain location. The information is complete because the values of each of the three variables representing unit position, category, and type are given, but it is imprecise because there is some ambiguity as to the exact type of the detected unit.
- **Uncertainty**: Uncertain information refers to cases where the information is complete, precise but uncertain since it might be wrong. This type of ignorance appears when the observer (human or sensor) is taken into account. It is the observer that is not certain about the available information.

Usually a database variable, i.e., column attribute for a relational database, is constrained either by a domain, when the variable is qualitative, or by a range, when the variable is numeric. The variables representing unit type and category are qualitative variables, whereas the variable representing unit position is a numeric one. Then it might be argued that incompleteness is just an extreme form imprecision. Moreover, imprecision and incompleteness are context dependent in the sense that information that is imprecise or incomplete in one context may be precise or complete in another context. For example, the imprecise information "tank or Humvee" is sufficient for a mission to target any hostile unit, but insufficient for destroying only those targets with striking capabilities. Similarly, if the mission is to target any hostile object in the air within the 100 miles radius of the base, then the information like "a hostile object is within 50 miles" or "a hostile object is very close" is sufficient to determine that the object is a target. But if the mission is to target any hostile object within the 40 miles radius of the base then neither piece of information is sufficient.

The following table presents subcategories of ignorance, example instances, and techniques for handling ignorance.

Table 4: Classification of Uncertainties

Type	Subcategory	Example	Formalism
Incompleteness	Existential	A hostile unit has been detected at a certain location, but its type is unknown	First-order logic
	Universal	All the units that have been detected so far at various locations are hostile, but their types are unknown	First-order logic
Imprecision	Disjunctive	The unit type is tank or armored personnel carrier	Classical logics
	Negation	The unit type is not tank	Non-monotonic logics
	Interval-valued	A tracked hostile unit is traveling between 25 and 40 mph	Interval theory
	Fuzzy-valued	A tracked unit is moving very fast	Fuzzy logic
	Possibility (physical)	The possibility that the traveling speed of the tracked hostile unit to be 30 mph	Modal epistemic logics, possibilistic logic, and possibility theory

Uncertainty	Probability	The chance of the unit being tank is 70%	Bayesian theory
	Possibility (epistemic)	It is possible that the tracked hostile unit is traveling at 30 mph	Modal epistemic logics, possibilistic logic, and possibility theory
	Credibility	0.8 is my degree of belief that the unit type is tank	Theory of belief function and certainty factor

The uncertainty handling problem can simply be stated as follows. Suppose a decision maker or an analyst has the universe of discourse $\Omega = \{w_1, ..., w_n\}$ at hand as the set of likely hypotheses for a given decision-making problem. Quantification of uncertainty involves attaching these hypotheses with some values, be they probabilities, possibility measures or any other qualitative terms suitably defined by the decision maker, reflecting the decision maker's degrees of beliefs in their likely occurrences based on the available evidence. Any formalism that wants to represent quantified beliefs of a decision-making agent has two components: the static component describes the agent's state of belief given the information available to the agent, and the dynamic component explains how to update beliefs given new pieces of information become available to the agent.

Existential and universal incompleteness in a knowledge base can be represented using quantifiers of first-order logic. Disjunction and negation types of imprecision can be represented using connectives of both propositional and first-order logics. However, classical logic is monotonic in nature, and thus requires some kind of rule for inferring negative information from a knowledge base. Various types of non-monotonic logics have been studied over the years, of which Reiter's default logic is the most prominent one. Intervals and fuzzy-valued information can be handled by interval theory and fuzzy logic respectively.

Two forms of possibility have been described to handle via possibility theory: the physical and the epistemic (Hacking 1975). The first form, 'possible that', is related to our state of knowledge and is called epistemic. The second form, 'possible for', deals with actual abilities independently of our knowledge about them. The distinction is similar to the one between the epistemic concept of probability (called here the credibility) and the traditional one based on chance. Possibilistic logic extends possibility theory with a richer representation of domain knowledge via logical sentences.

The probability theory is the predominant paradigm for handling uncertainty, especially Bayesian probability where interpretation of a probability value is degree of belief. The Possibility theory (Zadeh 1978), is a natural tool to model the imprecision and the uncertainty when they occur together in ordinary language such as, "It is likely that the vehicle is moving fast". The theory of belief function or evidence theory was initiated by Dempster and developed by (Shafer 1976). It uses two related measures, called belief and plausibility measures, as upper and lower probabilities. The probability and possibility measures are special cases of the belief and plausibility measures.

Table shows the formalisms that we have considered here for handling various types of ignorance, nature of their output quantifying uncertainties, and examples. The hypothesis set is $\{None, Slight, Long\}$ representing qualitative values for expressing the delay in arrival of a vehicle.

Table 5: Uncertainty output types from different formalisms

Formalism	Output Uncertainty	Example
Bayesian Probability	Probability distribution	*None*: 0.25, *Slight*: 0.20, *Long*: 0.55
Dempster-Shafer	Belief distribution among focal elements	{*None*}: 0.25, {*None*, *Slight*}: 0.40, …
Fuzzy Logic	Aggregated firing strength and crisp values	*None*: 0.30, *Slight*: 0.60, *Long*: 0.40 Arrival Delay = 2.5 hours
Possibility Theory	Possibility measure	*None*: 0, *Slight*: 0.20, *Long*: 0.55
Certainty Factor	Certainty factor values	*None*: -0.25, *Slight*: 0.20, *Long*: 0.55
Symbolic	Qualitative values	*None*: Unlikely, *Slight*: Probable, *Long*: Certain

Table 6 shows various approaches to transformation between formalisms. Here we clarify a few transformation instances. Dempster-Shafer theory of belief function is a generalization of

Bayesian probability, and, conversely, any Bayesian approximation of belief function, such as Transferable Belief Model, can do the reverse transformation. A special section provided below for transformation between probability and possibility approaches. The transformation between probability and certainty factor can be achieved by adjusting the scales [0,1] and [-1,1] appropriately. Any transformation between a user-defined symbolic dictionary for representing uncertainty and the rest of the formalisms has to be customized individually. Since a transformation exists between probability and Dempster-Shafer, a transformation between Dempster-Shafer and rest of the formalism can be defined based on the transformation between probability and rest of the formalism. The highlighted entries in the table are indicative of those transformations that are well-developed.

Table 6: Approaches To Transformation Between Formalisms

	Bayesian Probability	Dempster-Shafer	Fuzzy Logic	Possibilistic Theory	Certainty Factor	Symbolic Dictionary
Bayesian Probability	X	Generalization	Membership Degree Interpretation of Probabilities	Transformations based on Consistency Principles	Mapping from Probabilities to Certainty Factors	Probability-to-Symbolic Mapping
Dempster-Shafer	Bayesian Approximation (Transferable Belief Model)	X	Via Bayesian Approximation	Via Bayesian Approximation	Via Bayesian Approximation	Belief Value-to-Symbolic Mapping
Fuzzy Logic	Normalization of Membership Degrees	Via Normalization	X	Possibility Measure Interpretation of Membership Degrees	Mapping from Membership Degree to Certainty Factors	Membership Degree-to-Symbolic Mapping
Possibilistic Theory	Transformations based on Consistency Principles	Belief Interpretation of Possibility Measures	Membership Degree Interpretation of Possibility Measures	X	Mapping from Possibility Measures to Certainty Factors	Possibility Measure-to-Symbolic Mapping
Certainty Factor	Normalization	Via Normalization	Via Normalization	Via Normalization	X	Certainty Factor-to-Symbolic Mapping
Symbolic Dictionary	Symbolic-to-Probability Mapping	Symbolic-to-Belief Value Mapping	Symbolic-to-Membership Degree Mapping	Symbolic-to-Possibility Measure Mapping	Symbolic-to-Certainty Factor Mapping	X

8.1 Relating Probability and Possibility

The transformation between probability and possibility approaches to handling uncertainty has been studied by several researchers. Most of these studies examined consistency principles that must be preserved for the transformation, and devised equations. Here we introduce a few well-known ones. We consider the universe of discourse Ω is the set $\{w_1,...,w_n\}$ of singletons, and the probability and possibility distributions on Ω are $p=(p_1,...,p_n)$ and $\pi=(\pi_1,...,\pi_n)$

respectively. Table presents a comparison of characteristics between probability and possibility approaches.

Table 7: Characteristics Comparison Between Probability And Possibility Approaches

Probability	**Possibility**
Measure: Probability P	Measure: Possibility Π
Distribution: p	Distribution: π
$P(A) = \sum_{x \in A} p(x)$	$\Pi(A) = \sup_{x \in A} \pi(x)$
$P(A \cup B) = P(A) + P(B)$, if $A \cap B = \Phi$	$\Pi(A \cup B) = \max(\Pi(A), \Pi(B))$
$P(A \cap B) = P(A) \cdot P(B)$, if A and B are independent	$N(A \cap B) = \min(N(A), N(B))$
$\max(\Pi(A), \Pi(\overline{A})) = 1$ $\Pi(A) + \Pi(\overline{A}) \geq 1, N(A) + N(\overline{A}) \leq 1$	$P(A) + P(\overline{A}) = 1$
Ignorance: $\forall x \in \Omega, \pi(x) = 1$	Ignorance: $\forall x \in \Omega, p(x) = \frac{1}{\lvert\Omega\rvert}$

Zadeh (1978) stated that a high degree of possibility does not necessarily imply a high degree of probability, nor does a low degree of probability necessarily imply a low degree of possibility. He the defined the degree of consistency between a probability distribution and a possibility distribution as

$$C = \sum_{i=1}^{n} p_i \cdot \pi_i$$

This degree of consistency does not define a precise relationship between probability and possibility, but rather an approximate formalization of the heuristic connection that a lessening of the possibility of an event tends to lessen its probability but not vice-versa.

According to Klir (1993), the transformation from p_i and π_i must preserves some appropriate scale and the amount of information contained in each distribution expressed by the equality of their uncertainties. Assuming the elements of Ω are ordered in such a way that $p_i > 0, p_i \geq p_{i+1}$ and $\pi_i > 0, \pi_i \geq \pi_{i+1}$, where $p_{n+1} = \pi_{n+1} = 0$, Klir proposed the transformation under two scales satisfying his uncertainty preservation principle:

- Ratio scale: $\pi_i = \frac{p_i}{p_1}$, $p_i = \frac{\pi_i}{n \cdot \sum_{k=1}^{n} \pi_k}$
- Log-interval scale: $\pi_i = \left(\frac{p_i}{p_1}\right)^{\alpha}$, $p_i = \frac{\pi_i^{1,/\alpha}}{n \cdot \sum_{k=1}^{n} \pi_k^{1/\alpha}}$ $\quad$ *where* $\alpha \in (0,1)$

The consistency principle of Dubois and Prade (1993) is guided by the principle that values of probabilities of events are bounded by the degrees of possibility and necessity:

$$\Pi(A) \le P(A) \le N(A)$$

The transformation $\pi \to p$ below is guided by the principle of insufficient reason, which aims at finding the probability distribution that contains as much uncertainty as possible but that retains the features of possibility distribution:

$$p_i = \sum_{k=i}^{n} \frac{(\pi_k - \pi_{k+1})}{k}$$

The transformation $p \to \pi$ is guided by the principle of maximum specificity, which aims at finding the most informative possibility distribution:

$$\pi_i = \sum_{k=i}^{k} \min(p_i, p_k)$$

The two transformations defined above are shown to be bijective.

8.2 Non-monotonic Logic

Classical logic is monotonic in the sense that inferences are deductively valid and can never be retracted in the light of new information. On the other hand, non-monotonic logics support defeasible inferencing, the kinds of inference of everyday life in which conclusions are drawn tentatively, reserving the right to retract them in the light of further information. Thus the set of conclusions drawn on the basis of a given knowledge base does not necessarily increase with the size of the knowledge base itself. Default logic (Reiter, 1980) is probably the most extensively studied non-monotonic formalisms.

Default logic augments classical logic by default rules, and consequently the knowledge about the world is divided into two parts, representing respectively certain knowledge **W** in closed first-order formulae and a collection **D** of special inference rules called defaults. A default rule has the form

$$\frac{\alpha : \beta_1, ..., \beta_n}{\gamma}$$

where $\alpha, \beta_1, ..., \beta_n, \gamma$ are well-formed formulae of classical logic. α is called the prerequisite of the default, $\beta_1, ..., \beta_n$ are called justifications, and γ is the consequence.

A default rule is closed if none of $\alpha, \beta_1, ..., \beta_n, \gamma$ contains free variables. A default theory is closed if all its default rules are closed. A default or default theory is open if it is not closed. The semantics of a closed default theory $\langle \mathbf{D}, \mathbf{W} \rangle$ is based on the notion of extension, which is a possible state of the world according to the knowledge base. Formally, an extension E can be recursively defined as follows:

$$E_0 = W$$

$$E_{i+1} = Cons(E_i) \cup \left\{ \gamma : \frac{\alpha : \beta_1, ..., \beta_n}{\gamma} \in D, \alpha \in E_i, \forall j \neg \beta_j \in E_i \right\}$$

where $Cons(X)$ denotes classical deductive closure of X. The set $E = \bigcup_{i=0}^{\infty} E_i$ is deductively closed and an extension of $\langle \mathbf{D}, \mathbf{W} \rangle$.

The definition of extension is extended to open default theories by assuming that the default rules with free variables implicitly stand for the infinite set of closed defaults obtained by replacing the free variables with terms of the Herbrand Universe of the default theory. A default theory can have one, multiple or no extensions in general. Therefore, defining entailment (and thus semantics for negative information) of a formula from a default theory is not straightforward. The standard variants are credulous entailment under which a formula *F* is entailed if it belongs to at least one extension of $\langle \mathbf{D}, \mathbf{W} \rangle$ and skeptical entailment under which *F* follows if it belongs to all extensions of $\langle \mathbf{D}, \mathbf{W} \rangle$.

8.3 Certainty Factor

One of the early approaches to handling uncertainty in expert systems is Certainty Factors (Shortliffe and Buchanan, 1975). The formalism provides a number to measure the decision maker's degree of confirmation or belief in hypothesis, where the maximum value for belief is +1.0 (definitely true), and the minimum value is ˜-1.0 (definitely false). An intuitive interpretation of qualitative terms in terms of belief values are shown in the following table:

Table 8: Qualitative interpretations of certainty factors

Qualitative Term	Certainty Factor
Definitely	+1.0
Almost certainly	+0.8

Probably	+0.6
Maybe	+0.4
Unknown	[-0.2, +0.2]
Maybe not	-0.4
Probably not	-0.6
Almost certainly not	-0.8
Definitely not	-1.0

In the certainty factor formalism, unlike in the theory of probability, a piece of evidence *E* could support a hypothesis *H* without necessarily supporting the complement of that hypothesis. In other words, although the property

$$p\left(H \mid E\right) = 1 - p\left(\neg H \mid E\right)$$

holds in probability theory, but the certainty formalism rejects the following:

$$belief\left(H \mid E\right) = f\left(belief\left(\neg H \mid E\right)\right)$$

The approach creates the following measures of belief and disbelief:

$$MB(H,E) = \begin{cases} 1 & if\ p(H) = 1 \\ \dfrac{p(H,E) - p(H)}{1 - p(H)} & otherwise \end{cases}$$

$$MD(H,E) = \begin{cases} 1 & if\ p(H) = 0 \\ \dfrac{p(H) - p(H \mid E)}{p(H)} & otherwise \end{cases}$$

The value $1 - p(H)$ represents the decision maker's total doubts in the hypothesis, and thus the measure of belief *MB* based on evidence *E* computes the reduction in the decision maker's disbelief. Similarly, the measure of disbelief *MD* based on evidence *E* computes the reduction in decision maker's belief. The measure of belief is then computed as follows:

$$cf = \frac{MB - MD}{1 - \min(MB, MD)}$$

Different values of *cf* are interpreted as follows:

- *cf* is positive: evidence supports hypothesis since $MB > MD$
- *cf* is 1.0: evidence definitely supports the hypothesis
- *cf* is 0.0: either there is no evidence or that the belief is cancelled out by the disbelief
- *cf* is negative: evidence favors negation of hypothesis since $MB < MD$

Certainty factor assigned by a rule is propagated through the reasoning chain. It involves establishing the net certainty of the rule consequent when the evidence in the rule antecedent is uncertain. Given the rule

IF *E* THEN *H* (*cf*)

and given $CF(E)$, the combined belief in the hypothesis is computed by propagating the certainty factor as follows:

$$CF(H,E) = CF(E) \times cf$$

The following are the certainty factor propagation rules for conjunctions and disjunctions:

$$CF(E_1 \wedge ... \wedge E_n) = \max[CF(E_1),...,CF(E_n)]$$
$$CF(E_1 \vee ... \vee E_n) = \min[CF(E_1),...,CF(E_n)]$$

Finally, one needs to combine certainty factors for a hypothesis implicated by more than a rule as below:

IF *E*1 THEN *H* (*cf*1)

IF *E*2 THEN *H* (*cf*2)

The combination of *cf*1 and *cf*2 for *H* given both *E*1 and *E*2 is

$$CF(cf_1, cf_2) = \begin{cases} cf_1 + cf_2 \times (1 - cf_1) & \text{if } cf_1 > 0 \text{ and } cf_2 > 0 \\ \dfrac{cf_1 + cf_2}{1 - \min[|cf_1|, |cf_2|]} & \text{if } cf_1 < 0 \text{ or } cf_2 < 0 \\ cf_1 + cf_2 \times (1 + cf_1) & \text{if } cf_1 < 0 \text{ and } cf_2 < 0 \end{cases}$$

A natural requirement for any formalism for handling uncertainty is that the evidence combination rule should be associative. But this rule for the certainty factor formalism is not associative. The rule should never be applied if *E*1 and *E*2 are deduced from a common piece of evidence *E*0. To illustrate, suppose we have the following rules

IF *E*0 THEN *E*1 (1.0)

IF *E*0 THEN *E*2 (1.0)

IF *E*1 THEN *H* (1.0)

IF *E*2 THEN *H* (1.0)

Let the certainty factor of *E*0 be α, then both *E*1 and *E*2 have certainty factor equal to α. A blind application of the combination rule leads to a certainty factor of $2\alpha - \alpha^2$ for *H*. The result is wrong as the set of four rules can be reduced into

IF *E*0 THEN *H* (1.0)

and therefore the certainty factor of H given $E0$ should be α. This example shows the danger of using ad hoc models blindly.

8.4 Fuzzy Sets and Fuzzy Logic

A fuzzy set *A* is a subset of a referential set or universe of discourse *X* whose boundaries are gradual. More formally, the membership function μ_A of a fuzzy set *A* assigns to each element *x* in *X* its degree of membership $\mu_A(x)$ usually taking values in [0,1]. In a conventional set, $\mu_A(x)=1$ (means that *x* belongs to *A*) or $\mu_A(x)=0$ (means that *x* does not belong to *A*).

Consider the referential set as the interval $\left[-25^0,120^0\right]$ representing temperature values, and the three associated fuzzy sets Cold, Warm, and Hot. The membership functions μ_{Cold}, μ_{Warm}, and μ_{Hot} in terms of three trapezoidal graphs have been shown in Figure . Considering the first membership function, any temperature value of approximately less than 25 is assumed to have the degree of membership 1.0, and this degree gradually decreases as the temperature value increases. The degree of membership is 0 for any temperature value greater than 50. The other membership functions can be interpreted in a similar manner. Note that a membership function can be a probability density function, for example, μ_{Warm} is Gaussian with mean 75 and standard deviation 5.0.

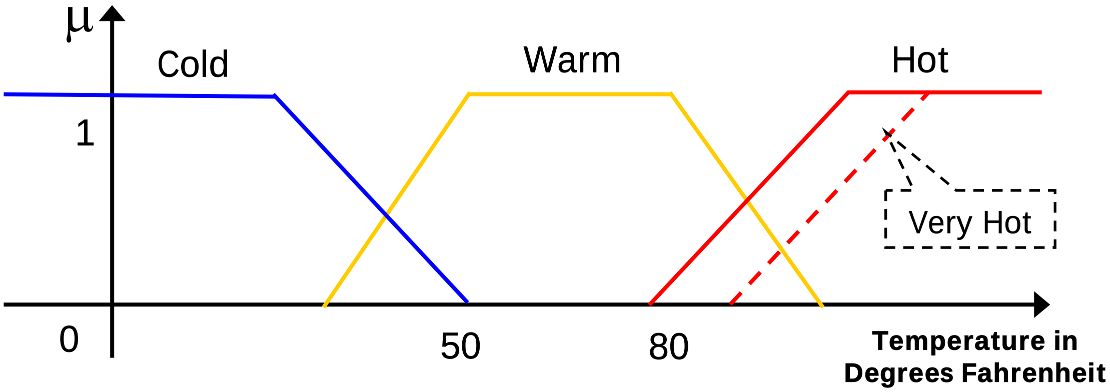
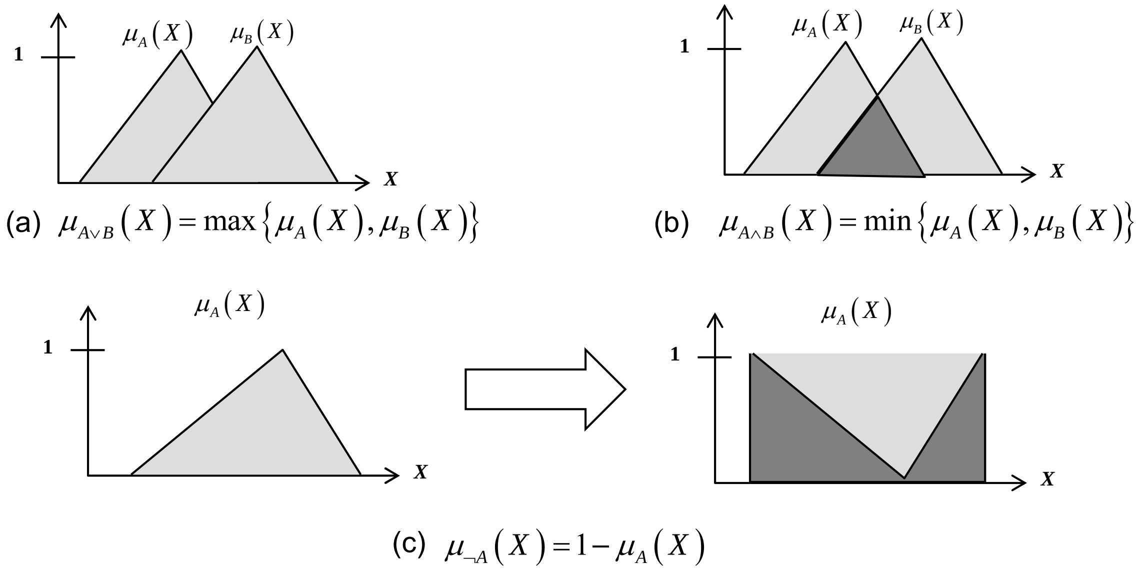


Figure 35: Fuzzy membership functions

The set theoretic operations on fuzzy sets are defined as below and illustrated in Figure :

- Union: $\mu_{A\vee B}(X)=\max\{\mu_A(X),\mu_B(X)\}$
- Intersection: $\mu_{A\wedge B}(X)=\min\{\mu_A(X),\mu_B(X)\}$
- Complement: $\mu_{\neg A}(X)=1-\mu_A(X)$



(a) $\mu_{A\vee B}(X)=\max\{\mu_A(X),\mu_B(X)\}$

(b) $\mu_{A\wedge B}(X)=\min\{\mu_A(X),\mu_B(X)\}$

(c) $\mu_{\neg A}(X)=1-\mu_A(X)$

Figure 36: Fuzzy set operations: (a) Union; (b) Intersection; and (c) Complement

Fuzzy logic provides a foundation for approximate reasoning by extending traditional Boolean logic and by allowing truth to be a matter of degree as in the case of membership degree in fuzzy sets. It is applicable to problems where the expressive power of words is greater than that of numbers. For example, a human observer reported that an object was approaching very fast, without knowing the exact speed of the object. Words like "fast" are less precise than numbers, but imprecision can be tolerated to achieve robustness. The four facets of fuzzy logic are shown in Figure .

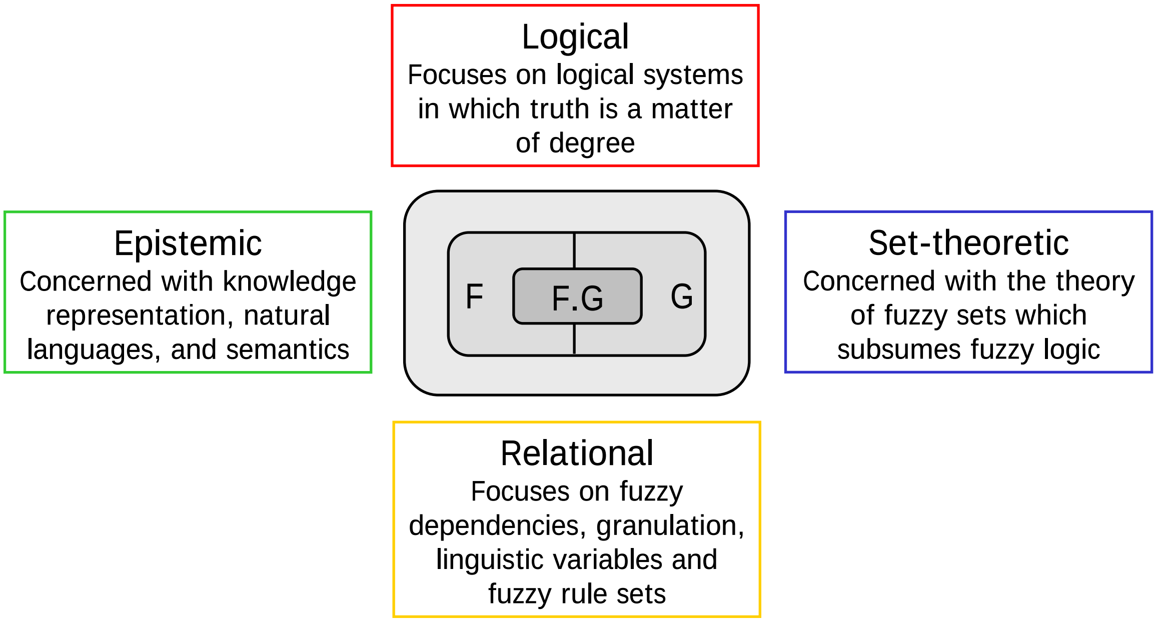

Logical
Focuses on logical systems in which truth is a matter of degree
Epistemic
Concerned with knowledge representation, natural languages, and semantics
F
F.G
G
Set-theoretic
Concerned with the theory of fuzzy sets which subsumes fuzzy logic
Relational
Focuses on fuzzy dependencies, granulation, linguistic variables and fuzzy rule sets

Figure 37: Facets of fuzzy logic (Zadeh, 2002)

Here is an example traditional logic-based exact reasoning just to contrast with fuzzy reasoning:

Exact Reasoning: IF engine temperature is hot THEN cooling fan speed is high
Engine temperature is hot
Therefore, cooling fan speed is high

In the case of fuzzy reasoning, the concept of a linguistic variable plays a central role. In the reformulation of fuzzy reasoning of the above exact reasoning, the linguistic variables are Temperature and Speed. Values of linguistic variables are words or sentences in a natural or synthetic language, and are constructed by composing primary terms or predicates with modifiers. In the case of the variable Temperature, primary terms or predicates are, for example, cold, warm, hot, …, and modifiers are, for example, not, very, quite, rather, …. Thus, linguistic values of the variable Temperature are hot, not cold, very cold, and so on. A numerical or "crisp" value of the variable Temperature is in its universe of discourse, which is the interval $\left[-25^0, 120^0\right]$. Modifiers fall into two classes:

- Fuzzy truth qualifier: quite true, very true, more or less true, etc.
- Fuzzy qualifier: many, few, almost

An example fuzzy proposition is "engine's temperature is very hot." Fuzzy expressions or formulae are built using fuzzy propositions, logical connectives, and quantifiers. Here is an example fuzzy formula:

IF rainfall is medium AND vehicle is heavy THEN delay is long

where the linguistic variable are Rainfall, Vehicle, and Delay. A fuzzy proposition can have its truth value in the interval [0, 1] defined by a membership function.

In proposition "*X* is *P*", *P*(*X*) (or simply *P* when *X* is clear from the context) is a fuzzy set and the membership function $\mu_P(X)$ (or simply μ_P) defines the degree of membership in the set. Truth value of the fuzzy predicate is defined by the membership function. Consider the three values, namely Cold, Warm, and Hot, of the linguistic variable Temperature. Then the truth value of the proposition "Temperature is Cold" is defined by μ_{Cold}.

A generic fuzzy reasoning architecture is shown in Figure . The Fuzzification component computes the membership degrees for each input variable with respect to its linguistic terms. The Fuzzy Reasoning component yields the output fuzzy set using computed membership degrees and the fuzzy rules, via rule matching, inference, and aggregation. Finally, the Defuzzification component determines a crisp value from the output membership function as the final result of solution

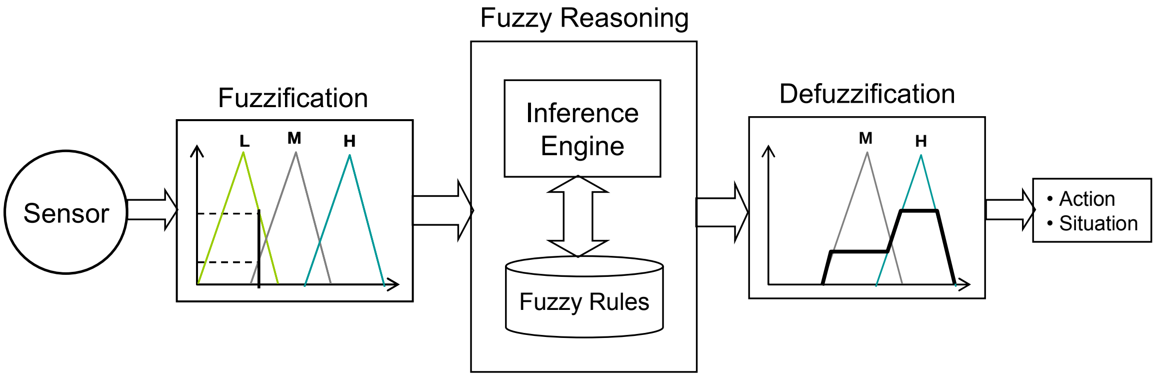


Figure 38: Generic fuzzy engine

We illustrate the fuzzy reasoning process in the context of an example scenario involving estimation of a vehicle's delay of arrival due to rainfall, which affects the mobility of the vehicle. There are three linguistic variables: Weight, Rainfall, and Delay. The underlying fuzzy rule base relating these variables is captured in Table 9. Each entry in the table is a value of the variable Delay given the value of the two Weight and Rainfall in the corresponding column and row respectively. For example, the shaded cell corresponds to the following rule:

IF weight is heavy AND rainfall is medium THEN delay is long

Table 9: Fuzzy Rule Base

		Vehicle Weight		
	Delay in Arrival	**Heavy**	**Medium**	**Light**
Rainfall	**Light**	Slight	None	None
	Medium	Long	Slight	None
	Heavy	Long	Long	Slight

The set of rules are as follows when traversing the cells row-by-row starting from left:

R1: IF weight is heavy AND rainfall is light THEN delay is slight

R2: IF weight is medium AND rainfall is light THEN delay is none

R3: IF weight is light AND rainfall is light THEN delay is none

R4: IF weight is heavy AND rainfall is medium THEN delay is long

R5: IF weight is medium AND rainfall is medium THEN delay is slight

R6: IF weight is light AND rainfall is medium THEN delay is none

R7: IF weight is heavy AND rainfall is heavy THEN delay is long

R8: IF weight is medium AND rainfall is heavy THEN delay is long

R9: IF weight is light AND rainfall is heavy THEN delay is slight

There will be altogether nine rules corresponding to the nine cells containing the values of the variable Delay. The membership functions for each of these variables are shown in Figure . Note that the membership function for $\mu_{Medium}(Rainfall)$ is Gaussian with mean 28.0 and variance is 6.0.

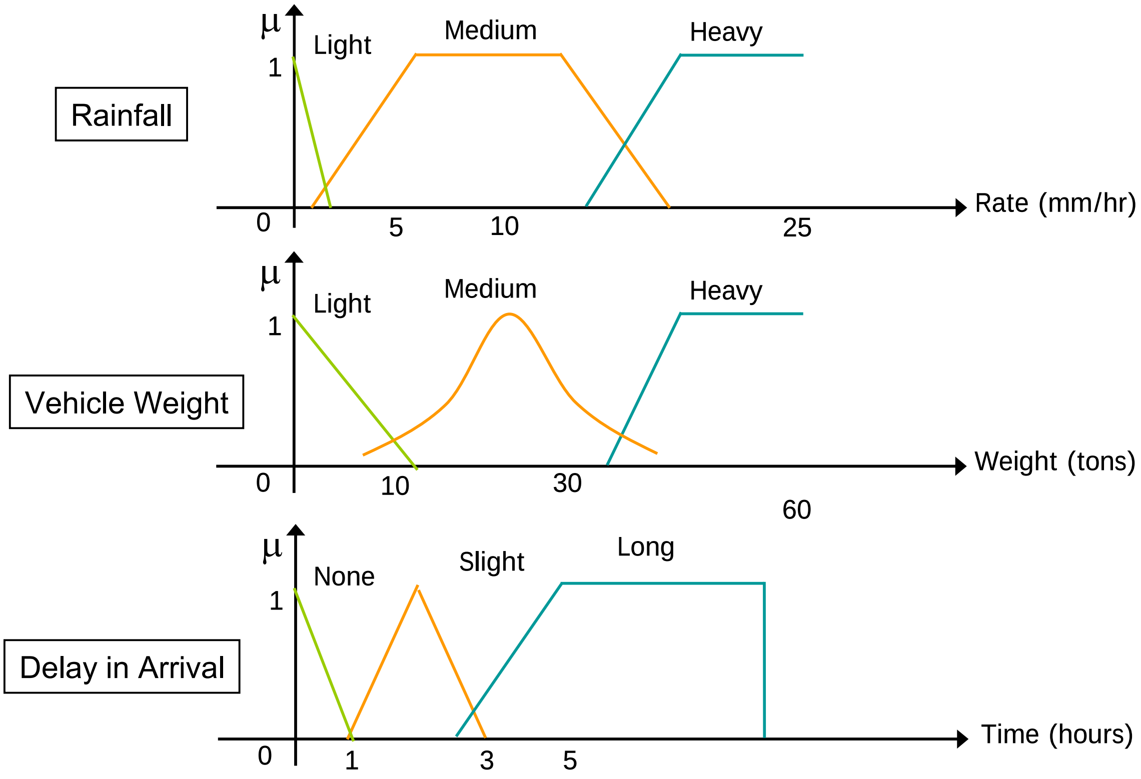


Figure 39: Fuzzy sets of the variables Rainfall, Weight, and Delay

In the fuzzification stage, one needs to determine, given the value of an input variable, the degree of membership for each term of the variable. Given the rainfall is 15.2 mm/hr and vehicle weight is 40 tons, Figure illustrates the determination of membership degrees and the resultant values are the following:

$\mu_{Light}(15.2)=0$ $\mu_{Light}(40)=0$

$\mu_{Medium}(15.2)=0.25$ $\mu_{Medium}(40)=0.1$

$\mu_{Heavy}(15.2)=0.70$ $\mu_{Heavy}(40)=0.60$

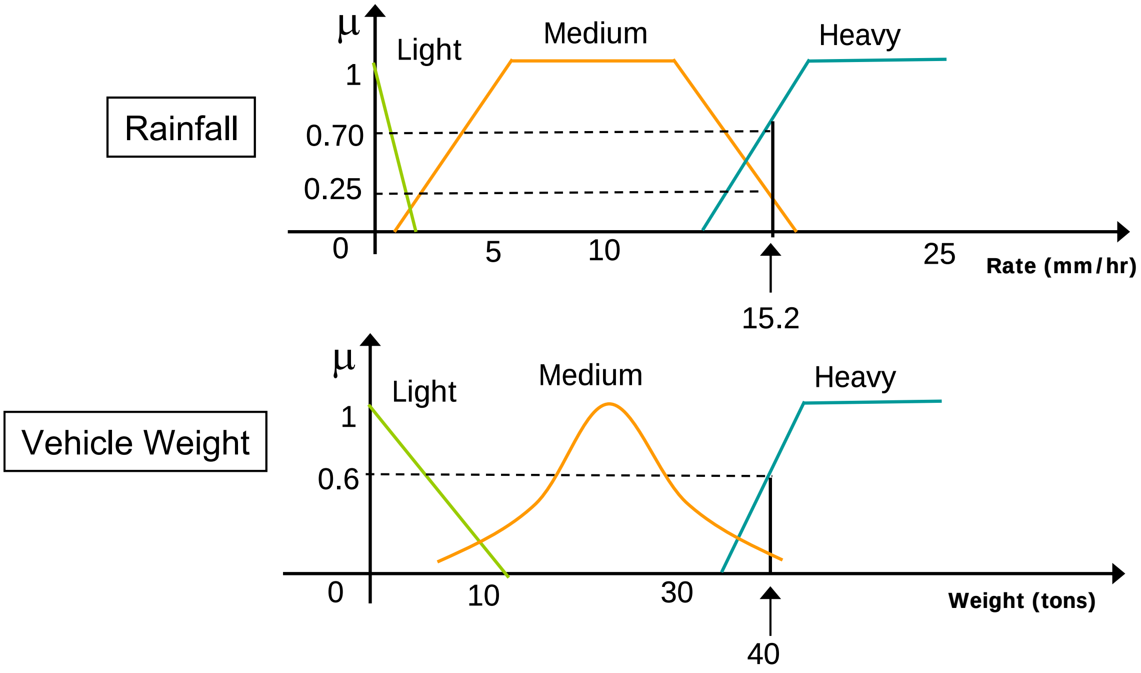


Figure 40: Determining degrees of membership for a value of the input variable

We then calculate the firing strength of every rule by combining the individual membership degree for all terms involved in the antecedent of a rule. Consider the rule

IF weight is heavy AND rainfall is medium THEN delay is long

Since we have $\mu_{Heavy}(Weight)=0.60$ and $\mu_{Medium}(Rainfall)=0.25$, and the antecedent of the rule is a conjunction, the combined membership degree is $\min\{\mu_{Heavy}(40),\mu_{Medium}(15.2)\}=0.25$.

The next step is, for each rule, to apply the firing strength to modify its consequent fuzzy set, resulting in a new fuzzy set as the result of applying the rule. Consider again the following rule as an example:

IF weight is heavy AND rainfall is medium THEN delay is long

Figure shows the modification of its consequent fuzzy set with the firing strength 0.25.

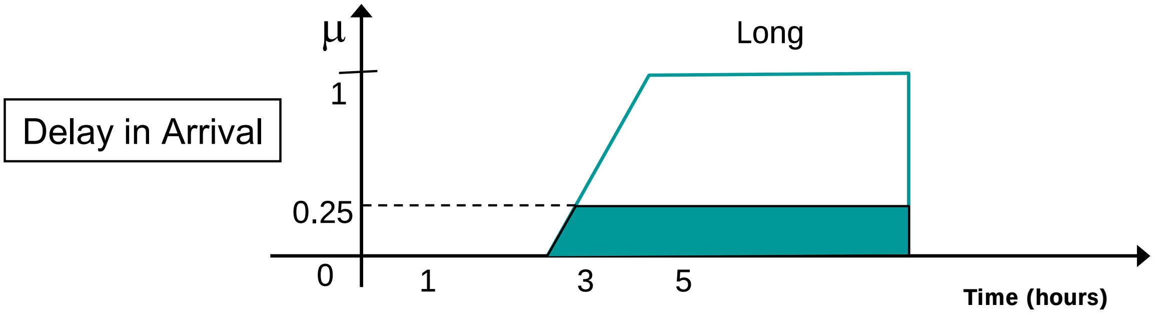


Figure 41: Modification of a rule's consequent fuzzy set

This modification is essentially reduces the degree of membership for every value of the variable Delay in the conclusion. What we are interested in is the shaded area underneath. There will be nine such modifications corresponding to the consequents of the nine rules in the fuzzy rule base, and all such modifications need to be aggregated. This involves combination of responses of individual rules to yield an overall output fuzzy set using the max-operator (i.e. superimpose the shaded areas). An example aggregation involving just the three types of modifications are shown in Figure .

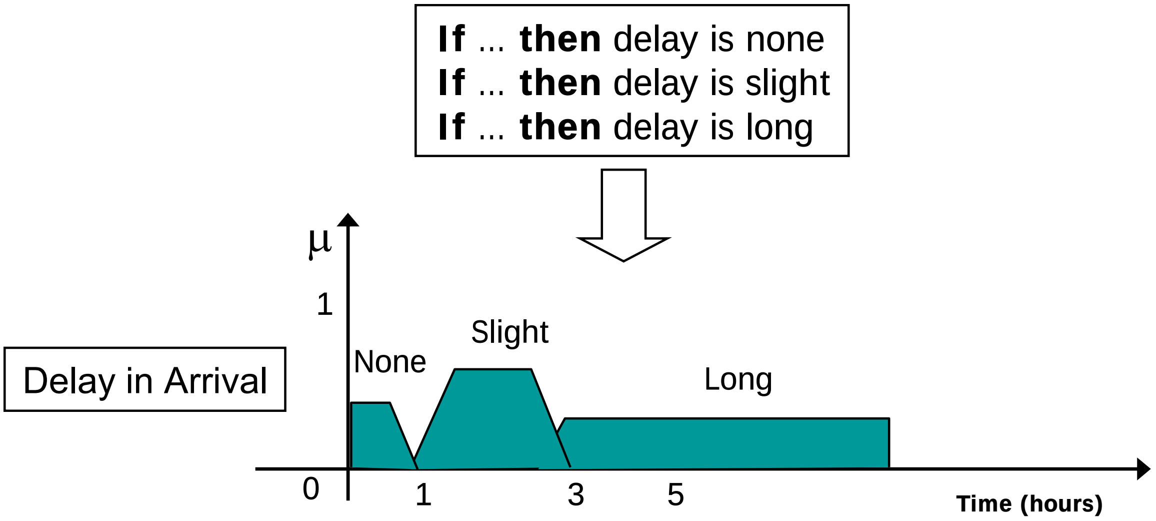

If ... **then** delay is none
If ... **then** delay is slight
If ... **then** delay is long
μ
1
Delay in Arrival
None
Slight
Long
0
1
3
5
Time (hours)

Figure 42: Aggregation of modified fuzzy sets

The final defuzzification process determines crisp value from output membership function. Two of the more common defuzzification techniques are the Centroid and Maximum methods. In the Centroid method, the crisp value of the output variable is computed by finding the variable value of the center of gravity of the membership function for the fuzzy value. In the Maximum method, one of the variable values at which the fuzzy subset has its maximum truth value is chosen as the crisp value for the output variable.

Figure 43 shows the projection of the center of gravity of the aggregated membership function onto the time-axis, yielding the crisp value 2.5 hours. This value is the estimated delay in arrival given the crisp values for vehicle weight and rainfall.

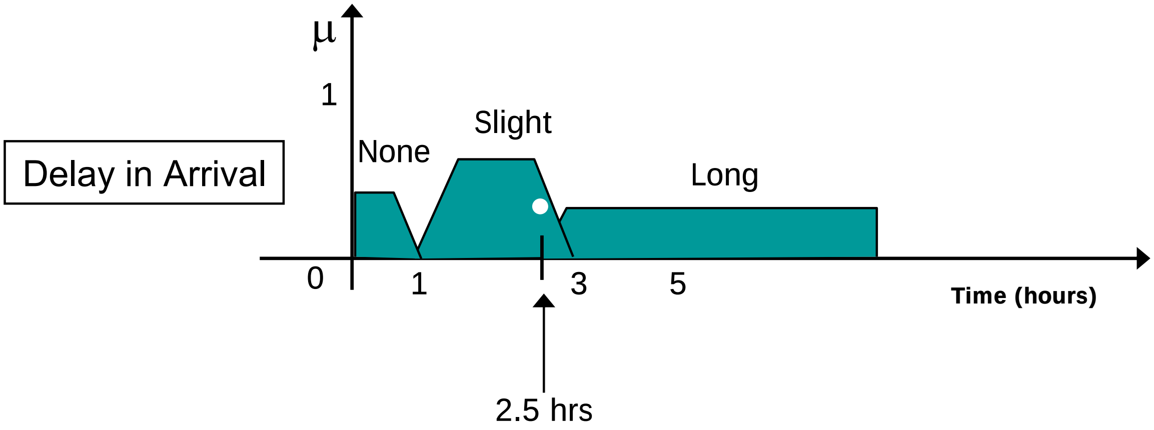


Figure 43: Defuzzification

8.5 Possibility Theory

Zadeh (1978) introduced possibility theory as an extension of fuzzy sets and fuzzy logic. The possibility of an event defined over a finite set of reference or universe of discourse Ω (an event is a subset of Ω) is a coefficient ranging between 0 and 1 which evaluates how possible the event is. Formally, the possibility is measured as a mapping given below:

$$\Pi : \mathcal{P}(\Omega) \to [0,1]$$

where $\mathcal{P}(\Omega)$ is the power set of Ω representing all possible events. The value of $\Pi(X)$ is 1 means the event X is completely possible and 0 means the events is impossible. The mapping must satisfy the following properties:

$$\Pi(\Phi) = 0$$
$$\Pi(\Omega) = 1$$
$$\Pi(A \cup B) = \max(\Pi(A), \Pi(B)) \quad \textit{where } A \cap B = \Phi$$

When Ω is infinite, the last axiom can be replaced by $\Pi\left(\bigcup_i A_i\right) = \sup_i \Pi(A_i)$, where A_i s are pairwise disjoint. Though possibility is compositional with respect to the union operator, it is not compositional with respect to the intersection operator. In general,

$$\Pi(A \cap B) \le \min(\Pi(A), \Pi(B))$$

The following properties of the possibility measure are deducible:

$$\max(\Pi(A), \Pi(\bar{A})) = 1$$
$$\Pi(A) + \Pi(\bar{A}) \ge 1$$

It is possible that each of $\Pi(A)$ and $\Pi(\overline{A})$ is 1 though only one of A and $\overline{A}$ can occur. The degree to which the occurrence of an event A is certain is represented by a necessity measure $N : \mathcal{P}(\Omega) \to [0,1]$ with the following properties:

$$\begin{aligned} &N(\Phi) = 0 \\ &N(\Omega) = 1 \\ &N(A \cap B) = \min(N(A), N(B)) \end{aligned}$$

When Ω is infinite, the last axiom can be replaced by $N\left(\bigcup_i A_i\right) = \inf_i N(A_i)$, where A_i s are pairwise disjoint. The necessity measure is monotonic:

$$if\ B \subseteq A\ then\ N(B) \le N(A)$$

The following properties of the necessity measure are deducible:

$$\begin{aligned} &N(A \cup B) \ge \max(N(A), N(B)) \\ &\min(N(A), N(\overline{A})) = 0 \\ &N(A) + N(\overline{A}) \le 1 \end{aligned}$$

The necessity measure can be obtained from the possibility measure via the following:

$$N(A) = 1 - \Pi(\overline{A})$$

Thus the occurrence of an event A is certain (i.e. $N(A) = 1$) if and only if the occurrence of its complement A is impossible (i.e. $\Pi(\overline{A}) = 0$).

Like probability, a possibility measure on Ω is determined by its behavior on singletons defined as a possibility distribution attached to a variable in Ω. A possibility distribution of a variable in Ω is defined as a mapping:

$$\pi : \Omega \to [0,1]$$

where $\pi(x) = 0$ means x is impossible, $\pi(x) = 1$ means x is completely allowed, and there exists $x \in \Omega$ such that $\pi(x) = 1$. The possibility measure $\Pi(A)$ is then defined as

$$\Pi(A) = \sup_{x \in A} \pi(x)$$

8.6 Possibilistic Logic

Possibilistic logic provides an approach to handling uncertainty in a logical setting. Standard possibilistic logic expressions are classical logic formulae associated with weights, interpreted in the framework of possibility theory as lower bounds of necessity degrees. A necessity measure N on formulas is a function from the set of logical formulas of a language to a totally ordered bounded scale with 0 and 1 as bottom and top elements, e.g., the real interval $[0,1]$, which is characterized by the axioms

- $N(\bullet\ )=1$, where • stands for tautology
- $N(\bot)=0$, where $\bot$ stands for contradiction
- $N(P\wedge Q)=\min(N(p),N(q))$
- $P\equiv Q$ implies $N(P)=N(Q)$, where $\equiv$ denotes equivalence in classical logic

A possibility measure Π is associated with *N* by duality as follows:

$$\Pi(P)=1-N(\neg P)$$

The above expresses that the absence of certainty in favor of $\neg P$ makes *P* possible. The following holds for all *P* and *Q*:

$$\Pi(P\vee Q)=\max(\Pi(P),\Pi(Q))$$

A first-order possibilistic logic formula is essentially a pair, (P,α), made of a classical first-order logic formula, *P*, and a weight, α, expressing certainty or priority. The axioms consist of all first-order axioms with weight 1. The inference rules are:

- If (P,α) and $(P\rightarrow Q,\beta)$ then $(Q,\min(\alpha,\beta))$
- If (P,α) then $(\forall x P,\alpha)$, provided the variable *x* is not bound in *P*
- If (P,α) and (P,β), where $\beta\leq\alpha$

From a semantic point of view, a possibilistic knowledge base $\mathbf{K}=\{(P_1,\alpha_1),\ldots,(P_n,\alpha_n)\}$ is understood as the possibility distribution $\pi_{\mathbf{K}}$ representing the fuzzy set of models of **K**:

$$\pi_{\mathbf{K}}(w)=\min_{1..n}\max(\mu_{[P_i]}(w),1-\alpha_i)$$

where $[P_i]$ denotes the set of models of P_i so that $\mu_{[P_i]}(w)=1$, if $w\in P_i$ (i.e. w ‘ P_i), and $\mu_{[P_i]}(w)=0$ otherwise. The degree of possibility of *w* according to the above is computed as the complement to 1 of the largest weight of a formula falsified by *w*. The model w^* maximizing $\pi_{\mathbf{K}}$ is called the best model of **K**. The best model is the most compatible with **K** among all the interpretations. The following equivalence establishes that knowing $\pi_{\mathbf{K}}$ is sufficient for all deductions from **K**:

$$\mathbf{K}(P,\alpha)\text{ if and only if }\pi_{\mathrm{K}}(P,\alpha)$$

An important feature of possibilistic logic is its ability to deal with inconsistency. The level of inconsistency of a possibilistic logic base is defined as

$$Inc(\mathbf{K})=\max\{\alpha{:}\mathbf{K}(\bot,\alpha)\}$$

More generally, $Inc(\mathbf{K})=0$ if an only if $\{P_i:(P_i,\alpha_i)\in\mathbf{K}\}$ is consistent in the classical sense. This would not be true in case α_i did represent a lower bound of the probability of P_i in a probabilistically weighted logic. The proposed system is sound and complete with respect to the inconsistency-tolerant semantics of possibilistic logic:

$$\mathbf{K}(F,\alpha) \text{ if and only if } \mathbf{K'}\ (F,\alpha), \text{ for any formula } F$$

The resolution principle for possibilistic logic is given as below:

$$(F_1,\ \alpha)\,(F_2,\ \beta)\,(\mathcal{R}(F_1,F_2),\ \min(\alpha,\beta))$$

where $\mathcal{R}(F_1,F_2)$ is the standard classical logic resolvent. The soundness of the rule can be established as follows. Let **K** be a set of possibilistic clauses, and (F,α) a possibilistic clause obtained by a finite number of successive applications of the resolution rule to **C**; then $\mathbf{K}\text{'}\ (F,\alpha)$.

Refutation can be easily extended to possibilistic logic. Let **K** be a knowledge base made of possibilistic formulas, i.e., $\mathbf{K} = \{(P_1,\alpha_1),...,(P_n,\alpha_n)\}$. Proving (F,α) from **K** amounts to adding $(\neg F,1)$, put in clausal form, to **K**, and using the above rules repeatedly until getting **K**, $(\neg F,1)\,(\bot,\alpha)$. We are interested here in getting the empty clause with the greatest possible weight. It holds that $\mathbf{K}(F,\alpha)$ if and only if $\mathbf{K}_\alpha F$ in the classical sense, where $\mathbf{K}_\alpha = \max\{F : (F,\beta) \in \mathbf{K} \text{ and } \beta \geq \alpha\}$.

Triangular norm operation can be applied for information fusion under possibilistic logic. To illustrate, consider the two possibilistic knowledge bases $\mathbf{K}_1 = \{(P_1,\alpha_1),...,(P_n,\alpha_n)\}$ and $\mathbf{K}_2 = \{(Q_1,\beta_1),...,(Q_m,\beta_m)\}$. Triangular norms are associative non decreasing symmetric operations *tn* such that $tn(1,\alpha) = \alpha$ and $tn(0,0) = 0$. The main triangular norms are min, product and $\max(0,\alpha+\beta-1)$. Let π_{tn} be the result of the combination of $\pi_{\mathbf{K}_1}$ and $\pi_{\mathbf{K}_2}$. based on the triangular norm operation *tn*. Then, *tn* is associated with the following possibilistic logic base:

$$\mathbf{K}_{tn} = \mathbf{K}_1 \cup \mathbf{K}_2 \cup \{(F_i \vee G_j, ct(\alpha_i,\beta_j)) : (F_i,\alpha_i) \in \mathbf{K}_1 \textit{ and } (G_j,\beta_j) \in \mathbf{K}_2\}$$

Note that if $tn = \min$ then $\mathbf{K}_{tn}$ is semantically equivalent to $\mathbf{K}_1 \cup \mathbf{K}_2$.

8.7 Dempster-Shafer Theory of Belief Functions

The theory of belief functions (Shafer, 1976), also known as Dempster-Shafer theory, is a generalization of the Bayesian theory of subjective probability (mainly by virtue of its explicit definition of the concept of ignorance) to combine accumulative evidence or to change prior opinions in the light of new evidence. Whereas the Bayesian theory requires probabilities for each question of interest, belief functions allow us to base degrees of belief for one question (for example, whether the game is on) on probabilities for a related question. Arthur P. Dempster set out the basic ideas of the theory (Dempster, 1966) and then Glenn Shafer developed the theory further (Shafer, 1976). Briefly, the theory may be summarized as follows.

Suppose expert X (for example, a weatherman, traffic cop, or grounds man) says that the game is not on due to heavy rain. The decision maker's subjective probabilities for expert X being reliable or unreliable are 0.7 and 0.3. Now, expert X's statement must be true if reliable, but not necessarily false if unreliable. The expert's testimony therefore justifies 0.7 "degrees of belief" that the game is not on, but only a zero (not 0.3) degree of belief that the game is on. The numbers 0.7 and 0 together constitute a belief function.

Suppose subjective probabilities were based on the decision maker's knowledge of the frequency with which experts like X are reliable witnesses. 70% of statements made would be true by reliable witnesses, *n*% would be true by unreliable ones, and $(30-n)\%$ would be false statements by unreliable witnesses. 0.7 and 0 are the lower bounds of true probabilities $(70+n)/100$ and $(30-n)/100$ respectively. Thus, a single belief function is always a consistent system of probability bounds, but may represent contradictory opinions from various experts. For example, consider the belief function 0.7 and 0 from expert X's opinion of the game not being on, and 0.8 and 0 from expert Y's opinion of the game being on. The lower bound of the true probability for the game not being on in the first case is 0.7, but the upper bound is 0.2 in the second case, yielding contradiction.

Let Ω be a finite set of mutually exclusive and exhaustive propositions, called the *frame-of-discernment*, about some problem domain ($\Omega = \{Cancelled, \neg Cancelled\}$ in our example decision making problem) and $\Pi(\Omega)$ is be the power set of Ω. A *basic probability assignment* (BPA) or *mass function* is the mapping

$$m : \Pi(\Omega) \rightarrow [0,1]$$

which is used to quantify the belief committed to a particular subset *A* of the frame of discernment given certain evidence. The probability number $m(A)$, the *mass* of *A*, says how much belief there is that some member of *A* is in fact the case, where

$$m(\Phi) = 0 \text{ and } \sum_{A \subseteq \Omega} m(A) = 1$$

The value 0 indicates no belief and the value 1 indicates total belief, and any values between these two limits indicate partial beliefs. If the probability number *p* for only a partial set *A* of hypotheses is known then the residual complementary probability number $1-p$ is assigned to the frame-of-discernment, thus allowing the representation of ignorance. A basic probability assignment *m* is Bayesian if $m(A) = 0$ for every non-singleton set *A*. For any set $A \subseteq \Omega$ for which $m(A) \neq 0$, *A* is called a *focal element*.

The measure of total belief committed to $A \subseteq \Omega$ can be obtained by computing the *belief function Bel* for $A \subseteq \Omega$ which simply adds the mass of all the subsets of *A*:

$$Bel(A)=\sum_{B\subseteq A} m(B)$$

A *single* belief function represents the lower limit of the true probability and the following plausibility function provides the upper limit of the probability:

$$Pl(A)=\sum_{B\cap A\neq\Phi} m(B)=1-Bel(A^c)$$

Mass can be recovered from belief function as follows:

$$m(B)=\sum_{A\subseteq B}(-1)^{|A-B|}Bel(A)$$

So there is a one-to-one correspondence between the two functions *m* and *Bel.* Two independent evidences expressed as two basic probability assignments m_1 and m_2 can be combined into a single joined basic assignment $m_{1,2}$ by Dempster's rule of combination:

$$m_{1,2}(A)=\begin{cases}\dfrac{\sum\limits_{B\cap C=A} m_1(B)m_2(C)}{1-\sum\limits_{B\cap C=\Phi} m_1(B)m_2(C)} & A\neq\Phi\\ 0 & A=\Phi\end{cases}$$

8.8 Transferable Belief Model

Transferable Belief Model (TBM) (Smets 1988; Smets and Kennes, 1994) describes a two-level mental model in order to distinguish between two aspects of beliefs, belief as weighted opinions, and belief for decision making. The two levels are: the credal level, where beliefs are entertained, and the pignistic level, where beliefs are used to make decisions (credal and pignistic derive from the Latin words "credo"', I believe and "pignus", a wage, a bet).

Beliefs can be entertained outside any decision context. TBM assumes that beliefs at the credal level are quantified by belief functions (Shafer 1976). At the credal level beliefs are quantified by belief functions. The credal level precedes the pignistic level in that at any time, beliefs are entertained (and revised) at the creedal level. The pignistic level appears only when a decision needs to be made. When a decision must be made, beliefs held at the credal level induce a probability measure at the pignistic level. That probability measure is then used to compute the expected utilities. To map belief functions onto probability functions, TBM uses a transformation that is called the pignistic transformation. Note that Bayesian formalisms do not consider the credal level. They usually argue that beliefs coexist with decisions.

Smett's pignistic transformation is as follows. Let *m* be the basic belief assignment on a space Ω. Then for every element *w* of Ω, the pignistic probability, denoted *BetP* to distinguish it from the subjective probability that would quantify the agent's beliefs according to the Bayesians, is

$$BetP(w) = \sum_{w:w \in A \subseteq \Omega} \frac{m(A)}{|A|}$$

where $|A|$ is the number of elements of Ω in A. If a person's belief at the credal level is already represented by a probability function, then the pignistic transformation just amounts to the identify operator.

8.9 Probability

Probabilities are defined in terms of likely outcomes of random experiments. A repetitive process, observation, or operation that determines the results of any one of a number of possible outcomes is called a *random experiment*. An *event* is an outcome of a random experiment. The set of all possible outcomes of an experiment is called the *sample space* or *event space*.

A *probability* provides a quantitative description of the likely occurrence of a particular event. The probability of an event *x*, denoted as $p(x)$, is conventionally expressed on a scale from 0 to 1, inclusive.

As defined above, an event consists of a single outcome in the sample space. Let us generalize this definition by calling it an *elementary* (or *simple event* or *atomic event*), and by defining a *compound event* as an event that consists of multiple simple events. In general, an event is either a simple event or a compound event. Set theory can be used to represent various relationships among events. In general, if *x* and *y* are two events (which may be either simple or compound) in the sample space *S* then:

- $x \cup y$ means either *x* or *y* occurs (or both occur)
- $x \cap y$ or *xy* means both *x* and *y* occur
- $x \subseteq y$ means if *x* occurs then so does *y*
- $\bar{x}$ means event *x* does not occur (or equivalently, the complement of x occurs)
- Φ represents an impossible event
- *S* is an event that is *certain* to occur

Two events *x* and *y* are said to be *mutually exclusive* if $x \cap y = \Phi$. (The occurrence of both *x* and *y* is impossible, and therefore the two events are mutually exclusive.) On the other hand, two events *x* and *y* are said to be *independent* if $p(x \cap y) = p(x) \times p(y)$. As a result, when dealing with independent events *x* and *y* in an event space, the sets *x* and *y* must have a point (event) in common if both *x* and *y* have nonzero probabilities. Mutually exclusive, non-impossible events *x* and *y* cannot be independent as $x \cap y = \Phi$, so that $p(x \cap y) = 0$, but $p(x) \times p(y) \neq 0$.

There are three approaches that provide guidelines on how to assign probability values:

- The classical approach
- The relative frequency approach
- The axiomatic approach

In the *classical approach*, the probability of an event *x* in a finite sample space *S* is defined as follows:

$$p(x) = \frac{n(x)}{n(S)}$$

where $n(X)$ is the cardinality of the (finite) set *X*. Since $x \subseteq S$, $0 \le p(x) \le 1$ and $p(S) = 1$.

In the *relative frequency* approach, the probability of an event *x* is defined as the ratio of the number (say, *n*) of outcomes or occurrences of *x* to the total number (say, *N*) of trials in a random experiment. The choice of *N* depends on the particular experiment, but if an experiment is repeated at least *N* times without changing the experimental conditions, then the relative frequency of any particular event will (in theory) eventually settle down to some value. The probability of the event can then be defined as the limiting value of the relative frequency:

$$p(x) = \lim_{n \to \infty} \frac{n}{N}$$

where *n* is the number of occurrences of *x* and *N* is total number of trials. For example, if a die is rolled many times then the relative frequency of the event "six" will settle down to a value of approximately 1/6.

In the *axiomatic* approach, the concept of probability is axiomized as follows:

- $p(x) \ge 0$, where *x* is an arbitrary event
- $p(S) = 1$, where *S* is a certain event, i.e. the whole event space
- $p(x \cup y) = p(x) + p(y)$, where *x* and *y* are mutually exclusive events.

Note that while the axiomatic approach merely provides guidance on how to assign values to probabilities, the classical and relative frequency approaches specify what values to assign.

A *subjective probability* describes an individual's personal judgment about how likely a particular event is to occur. It is not based on any precise computation, but is an assessment by a subject matter expert based on his or her experience (that is, it's a "guesstimate").

Now we turn to formally defining random variables and probability distributions, the concepts central to the development of probabilistic models for decision-making. A *random variable* is a function defined over an event space (that is, the domain of a random variable consists of random events from the sample space) and its value is determined by the outcome of an event. A *discrete random variable* is a random variable whose range is finite or denumerable. The elements in the range, i.e., possible values, of a random variable are called its *states*.

The *probability distribution* of a random variable is a function whose domain is the range of the random variable, and whose range is a set of values associated with the probabilities of the elements of the domain. The probability distribution of a discrete random variable is called a *discrete probability distribution*. The probability distribution of a continuous random variable is called a *continuous probability distribution*. In this book, we are mainly concerned about discrete probability distributions.

The probability distribution of a discrete random variable is represented by its *probability mass function* (or *probability density function* or *pdf*). A probability density function *f* of a random variable *X* with states $\{x_1,...,x_n\}$ is defined as follows: $f(x_i)$ is the probability that *X* will assume the value x_i.

The *joint probability distribution* of two discrete random variables *X* and *Y*, denoted as $p(X,Y)$ or $p(XY)$, is a function whose domain is the set of ordered pairs (x,y) of events, where *x* and *y* are possible values for *X* and *Y*, respectively, and whose range is the set of probability values corresponding to the ordered pairs in its domain. Such a probability is denoted by $p(X=x,Y=y)$ (or simply $p(x,y)$ when *X* and *Y* are clear from the context) and is defined as

$$p(x,y)=p(X=x,Y=y)=p(X=x\,\&\,Y=y)$$

The definition of the joint probability distribution can be extended to three or more random variables. In general, the joint probability distribution of the set of discrete random variables $X_1,...,X_n$, denoted as $p(X_1,...,X_n)$ or $p(X_1...X_n)$, is given by

$$p(x_1,...,x_n)=p(X_1=x_1,...,X_n=x_n)=p(X_1=x_1\,\&\,...\,\&\,X_n=x_n)$$

The notion of conditional probability distribution arises when you want to know the probability of an event, given the occurrence of another event. For example, the probability of snowy weather later today given that the current temperature is freezing. Formally, the *conditional probability distribution* of the two random variables *X* and *Y*, denoted as $p(X\,|\,Y)$, is a function whose domain is the set of ordered pairs (x,y), where *x* and *y* are possible values for *X* and *Y*, respectively, and is a function whose range is the set of probability values corresponding to the ordered pairs. The conditional probability distribution is defined as follows:

$$p(X\,|\,Y)=\frac{p(XY)}{p(Y)},\ \text{if}\ p(Y)>0$$

In the Bayesian approach, the probability on a particular statement regarding random variable states is not based on any precise computation, but describes an individual's personal judgment about how likely a particular event is to occur based on experience. The Bayesian approach is more general and expected to provide better results in practice than frequency probabilities alone because it incorporates subjective probability. The Bayesian approach can therefore obtain different probabilities for any particular statement by incorporating prior information from experts. The Bayesian approach is especially useful in situations where no historic data exists to compute prior probabilities. Bayes' rule allows one to manipulate conditional probabilities as follows:

$$p(A \mid B) = \frac{p(B \mid A)p(A)}{p(B)}$$

This rule estimates the probability of A in light of the observation B. Even if B did not happen, one can estimate the probability $p(A \mid B)$ that takes into account the prior probability $p(A)$.

9. References

Al-Halimi, A., Berwick, R. C., Burg, J. F. M., Chodorow, M., Fellbaum, C., Grabowski, J. et al. (1998). *WordNet: An Electronic Lexical Database.*, Fellbaum, C. (Eds.). Cambridge, MA: MIT Press.

Bonissone, P. and Tong, R. (1985). Editorial: Reasoning with Uncertainty in Expert Systems. International Journal of Man-Machine Studies, pp. 241-250.

Canas, A., Hill, G., Carff, R., Suri, Lott, J., Eskridge, T. C. et al. (2004). CmapTools: A Knowledge Modeling and Sharing Environment. In *Proceedings of First International Conference on Concept Mapping,* (pp. 125-133). Pamplona, Spain: Universidad Pública de Navarra.

Carozzoni, J. (2005). *Commander's Predictive Environment.*

Carpenter, G. A., Gjaja, M. N., Gopal, S., & Woodcock, C. E. (1996). ART Neural Networks for Remote Sensing: Vegetation Classification From Landstat TM and Terrain Data. *IEEE Transactions on Geoscience and Remote Sensing*.

Das, S. K. (2005). Symbolic Argumentation for Decision Making Under Uncertainty. In *Proceedings of 8th International Conference on Information Fusion*. Philadelphia, PA.

Dubois D, Prade H (1993). On possibility/probability transformations, Fuzzy Logic : 103-112.

Gruber, T. R. (1993). Toward Principles for the Design of Ontologies Used for Knowledge Sharing. In *Proceedings of International Workshop on Formal Ontology*.

Holi, M. & Hyvönen, E. (2004). Probabilistic Information Retrieval Based on Conceptual Overlap in Semantic Web Ontologies. In *Proceedings of 11th Finnish AI Conference*. Finnish AI Society.

Jensen, F. V. (1996). *Bayesian Networks Basics.* Aalborg University, Denmark.

Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research,* 4.

Klir GJ (1993). Information-preserving probability-possibility transformations: recent developments, Fuzzy Logic: 417-428.

Kohonen, T. (1984). *Self-Organization and Associative Memory*. Berlin: Springer, Verlag.

Lenat, D. & Guha, R. V. (1990). *Building Large Knowledge-Based Systems: Representations and Inference in the Cyc Project*. Reading: Addison-Wesley.

Novak, J. D. (1998). *Learning, Creating, and Using Knowledge: Concept Maps As Facilitative Tools in Schools and Corporations*. Mahweh, NJ: Lawrence Erlbaum Associates.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA: Morgan Kaufmann.

Reiter, R. (1980). “A logic for default reasoning”. Artificial Intelligence, Vol. 13, pp. 81-132.

Robbins, J. D., Deckro, R. F., & Wiley, V. D. (2005). Stabilization and Reconstruction Operations Model (SROM). In *Proceedings of 4th Workshop on Critical Issues in Information Fusion*. New York.

Shafer, G. (1976). “Mathematical Theory of Evidence”. Princeton, NJ: Princeton University Press.

Shortliffe, E. H. (1976). “Computer-based medical consultations: MYCIN”. New York: American Elsevier.

Smets, P. (1991). “Varieties of ignorance and the need for well-founded theories,” Information Sciences, 57-58: 135-144.

Smets, P. and Kennes, R. “The transferable belief model,” Artificial Intelligence, vol 66, pp 191-234, 1994.

Sowa, J. F. (1991). *Principles of Semantic Networks: Exploration in the Representation of Knowledge*. San Mateo, CA: Morgan Kaufmann.

Zadeh, L. A. (1965). “Fuzzy Sets.” Information and Control 8: 338-353.

Zadeh, Lotfi, "Fuzzy Sets as the Basis for a Theory of Possibility", Fuzzy Sets and Systems 1:3-28, 1978.